AD-A007 562

CHECKING PROOFS IN THE METAMATHEMATICS OF FIRST ORDER LOGIC

Mario Aiello, et al

Stanford University

Prepared for:

Office of Naval Research
Advanced Research Projects Agency

August 1974

Stanford Artificial Intelligence Laboratory
Memo AIM-222

AUGUST 1974

Computer Science Department
Report No. STAN-CS-74-467

Checking Proofs in the Metamathematics of First Order Logic

by

Mario Aiello
Richard W. Weyhrauch

Research sponsored by

Advanced Research Projects Agency
ARPA Order No. 2494

COMPUTER SCIENCE DEPARTMENT
Stanford University

37

ARTIFICIAL INTELLIGENCE LABORATORY
MEMO AIM No.222

AUGUST 1974

STAN-CS-74-467

CHECKING PROOFS IN THE METAMATHEMATICS OF FIRST ORDER LOGIC

by
Mario Aiello
and
Richard W. Weyhrauch

Abstract:

This is a report on some of the first experiments of any size carried out using the new first order proof checker FOL. We present two different first order axiomatizations of the metamathematics of the logic which FOL itself checks and show several proofs using each one. The difference between the axiomatizations is that one defines the metamathematics in a many sorted logic, the other does not.

Authors' addresses

M. Aiello, Istituto di Scienze dell'Informazione, Universita' di Pisa, corso Italia 40, 56100 Pisa, Italy;

R. Weyhrauch, A.I. Lab., Computer Science Dept., Stanford University, Stanford, California 94305. or Weyhrauch @SU-AI

This research is supported (in part) by the Advanced Research Project Agency of the Office of the Secretary of Defense (DAHC 15-73-C-0435).

Checking metamathematical proofs

TABLE OF CONTENTS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: STAN-CS-74-467	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): CHECKING PROOFS IN THE METAMATHEMATICS OF FIRST ORDER LOGIC.	5. TYPE OF REPORT & PERIOD COVERED: technical, August, 1974
	6. PERFORMING ORG. REPORT NUMBER: STAN-CS-74-467
7. AUTHOR(s): M. Aiello and R. W. Weyhrauch	8. CONTRACT OR GRANT NUMBER(s): DAHC 15-73-C-0435
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Stanford University, Computer Science Department, Stanford, California 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: ARPA/IPT, Attn: Stephen D. Crocker, 1400 Wilson Blvd., Arlington, Va. 22209	12. REPORT DATE: August 1974
	13. NUMBER OF PAGES: 57
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): ONR Representative: Philip Surra, Durand Aeronautics Bldg., Rm. 165, Stanford University, Stanford, California 94305	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Releasable without limitations on dissemination.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This is a report on some of the first experiments of any size carried out using the new first order proof checker FOL. We present two different first order axiomatizations of the metamathematics of the logic which FOL itself checks and show several proofs using each one. The difference between the axiomatizations is that one defines the metamathematics in a many sorted logic, the other does not.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

PRICES SUBJECT TO CHANGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

v

SECTION 1 INTRODUCTION

This paper represents a first attempt at the axiomatization of the metamathematics of a first order theory and at using the new proof checker FOL (First Order Logic). The logic which FOL checks is described in detail in the user manual for this program, Weyhrauch and Thomas 1974. It is based on a system of natural deduction described in Prawitz 1965, 1970.

Our motivation in axiomatizing the metamathematics of FOL was the desire to work on an example which could be used as a case study for projected features of FOL and, at the same time, had independent interest with respect to representing the proofs of significant mathematical results to a computer.

The eventual ability to clearly express the theorems of mathematics to a computer will require the facility to state and prove theorems of metamathematics. There are several clear examples:

a. *Axiom schemas.* How exactly do we express that

$$P(0) \land \forall n.(P(n) \supset P(n+1)) \supset \forall n.P(n)$$

is an axiom schema? We need to say: "If for any first order sentence P with one free variable y we denote by P(n) the formula obtained from P by substituting n for y assuming n is free for y in P, then the sentence

$$P(0) \land \forall n.(P(n) \supset P(n+1)) \supset \forall n.P(n)$$

is an axiom of arithmetic".

b. *Theorem schemas.* The following kind of "theorem" is sometimes seen in set theory books

$$\forall x1 \ldots xn\ S.\ \exists T.\ \forall u.\ (\langle x1,\ldots,xn\rangle \in T \equiv \exists y.(\langle x1,\ldots,xn,y\rangle \in S)).$$

It asserts the existence of some particular projection of n+1-tuples. In its usual formulation this is not a theorem of set theory at all, but a metatheorem which states that, for each n, the above sentence is a theorem. We do not know of any implementation of first order logic capable of expressing the above notion in a straightforward way.

c. *Subsidiary deduction rules.* Below we show how to prove that if there is a proof of ∀x y.WFF then there is also a proof of ∀y x.WFF, where WFF is any well formed formula. We chose this task because it seemed simple enough to do, and is a theorem which may actually be used. The use of metatheorems as rules of inference by means of a reflection principle will be discussed in a future memo by Richard Weyhrauch. Eventually we hope to check some more substantial metamathematical theorems.

d. *Interesting mathematical theorems.* We present two examples. The first is any theorem about finite groups. The notion of finite group cannot be defined in the usual first order language of group theory. Thus many "theorems" are actually metatheorems, unless you axiomatize groups in set theory. The second theorem is the "duality principle" in projective geometry.

SECTION 2 THE AXIOM SYSTEM

In this section we present two axiomatizations of the metamathematics of first order logic. The main difference between them is that one is done in a many sorted first order logic and the other not. These axiomatizations represent an attempt at experimenting with proofs about properties of formulas and deductions. No effort has been spent on guaranteeing that the axioms are independent. It would not only have been uninteresting but also contrary to our basic philosophy. We wish to find axioms which naturally reflect the relevant notions. At the moment this axiomatization is far from being in its final form. Neither the extent of the notions involved nor the best way of expressing them is considered settled.

Section 2.1 The sorts

The sorts we have defined correspond to the basic notions of the metamathematics i.e. terms, formulas, individual variables, logical symbols, function symbols etc. and to the notions of the domains (strings and sequences of strings) in which the axiomatization has been defined. FOL (see Weyhrauch and Thomas 1974) allows the declaration of variables to be of a certain sort. In the formulas appearing in this paper the following declarations are assumed:

g g1 g2 g3 g4 g5 g6	range over the most general sort
sq sq1 sq2 sq3 sq4 sq5 sq6 ϵ SEQ	(SEQs are sequences of strings)
pf pf1 pf2 pf3 pf4 pf5 pf6 ϵ PROOFTREE	(PROOFTREEs are sequences representing derivations in FOL)
s s1 s2 s3 s4 s5 s6 ϵ STRING	(STRINGs are strings)
t t1 t2 t3 t4 t5 t6 ϵ TERM	(TERMs are strings representing terms)
x x1 x2 x3 x4 x5 x6 ϵ INDVAR	(INDVARs are strings representing individual variables)
el el1 el2 el3 el4 el5 el6 ϵ ELF	(ELFs are strings representing elementary formulas)
f f1 f2 f3 f4 f5 f6 ϵ FORM	(FORMs are well formed formulas)
th th1 th2 th3 th4 th5 th6 ϵ BEW	(BEWs are theorems of a first order theory)
A A1 A2 A3 A4 A5 A6 ϵ AXIOM	(AXIOMs are axioms of a particular theory)
c0 c1 c2 c3 c4 c5 c6 ϵ INDCONST	(INDCONSTs are individual constants)
a a1 a2 a3 a4 a5 a6 ϵ ATOM	(ATOMs are the individual constituents of a string)
n n1 n2 n3 n4 n5 k ϵ INTEGER	(INTEGERs are integers)
nc nc1 nc2 nc3 nc4 nc5 nc6 ϵ NUMERAL	(NUMERALs are numerals)

sy sy1 sy2 sy3 sy4 sy5 sy6 ∈ SYM (SYMs are logical symbols)

np np1 np2 np3 np4 np5 np6 ∈ N_PLCSYM (N_PLCSYMs are symbols which have an *arity*)

fn fn1 fn2 fn3 fn4 fn5 fn6 ∈ OPCONST, (OPCONSTs are function symbols)

P P1 P2 P3 P4 P5 P6 ∈ PREDCONST; PREDCONSTs are predicate symbols)

the partial order between these sorts is defined by the following FOL declarations:

```
MG SEQ          ≥ { STRING , PROOFTREE } ;
MG PROOFTREE    ≥ { FORM };
MG STRING       ≥ { TERM , FORM , ATOM , VARSTRING } ;
MG TERM         ≥ { INDVAR } ;
MG FORM         ≥ { ELF , SENTCONST , PREDPAR0 , AXIOM , BEW } ;
MG BEW          ≥ { AXIOM } ;
MG ATOM         ≥ { INDCONST , SENTCONST , SYM , INTEGER , N_PLCSYM ,
                    INDPAR , INDVAR , AUXSIGN , PREDCONST0 , PREDPAR0 };
MG INDCONST     ≥ { NUMERAL } ;
MG SYM          ≥ { QUANT , SENTCONN } ;
MG N_PLCSYM     ≥ { PREDCONST , OPCONST , PREDPAR } ;
```

Sorts are always predicates with one argument. The declaration

MG SORT1 ≥ { SORT2 , .. SORTn }

should be read as SORT1 is more general than SORT2,...,SORTn and corresponds to the implicit axioms

∀g.SORT1(g)⊃SORT1(g),

.

.

∀g.SORTn(g)⊃SORT1(g).

The first declaration, for instance, says that strings and derivations are particular sequences of formulas. Strings are in fact sequences of length 1 and derivations are those sequences satisfying the predicate PROOFTREE.

Section 2.2 The domain of representation of the metamathematics

The basic notions of the metamathematics of first order logic have been axiomatized in terms of strings and sequences of strings. The primitive functions on them are concatenation (c for strings, cc for sequences) and selectors (car, cdr for strings and scar, scdr for sequences). c and cc are infix operators.

2.2.1 Formulas and terms

Formulas and terms are represented by the string of symbols appearing in them. Terms are defined recursively as strings which either represent an individual variable or can be decomposed into $n+1$ substrings representing a function symbol of arity n, followed by n terms. The two predicates defining terms are:

TERMSEQ(0,LAMBDA)

∀s.(TERM(s) ≡ INDVAR(s) ∨ ∃n fn.(fn=car(s) ∧ n=arity(fn) ∧ TERMSEQ(n,cdr(s))))

∀n s.(TERMSEQ(n,s) ≡ ((car(s)=LPARSYM) ∧ ((len(s) gl s)=RPARSYM) ∧
∃n1.(TERM(substring(s,2,n1)) ∧ TERMSEQ(n-1,substring(s,n1+1,len(s)-1)))))

where the function substring(s,m,n) (see appendix 1.2) returns the substring of **s** starting from its m-th element and ending with the n-th. **len(s)** computes the length of **s** and **(n gl s)** selects the n-th element of **s**.

Well formed formulas (wffs) are represented as strings which either are elementary formulas (defined by the predicate ELF) or can be partitioned into substrings for formulas and logical connectives. Formulas are defined by:

∀s.(ELF(s) ≡ (s=FALSESYM ∨ PREDPAR0(s) ∨ ∃n P.(P=car(s) ∧ n=arily(P) ∧ TERMSEQ(n,cdr(s))))),

∀s.(FORM(s) ≡ (ELF(s) ∨ ∃x f.(s=(x gen f) ∨ s=(x ex f)) ∨
∃f1 f2.(s=(f1 dis f2) ∨ s=(f1 con f2) ∨ s=(f1 impl f2)) ∨ ∃f.s=neg(f))) ;;

gen is the infix operator that maps its arguments x and f into the string **(FORALLSYM c x) c f** representing the well formed formula ∀x.f. The operator **ex** is used for the existential quantifier. **dis**, **con** and **impl** are the infix operators for the disjunction, conjunction and implication of two formulas. Finally, **neg** is the operator which maps a formula into its negation.

We could possibly represent wffs as structured objects (lists, trees, etc.) which contain all the information about the structure of the formula and do not require any parsing. This approach amounts to axiomatizing metamathematics in terms of the abstract syntax of first order logic, instead of strings of symbols. Both of these possibilities should be explored. We have chosen the first alternative because:

1) It is the most traditional, i.e. metamathematics, as it appears in logic books, is usually stated in terms of strings.

2) Axioms in terms of abstract syntax are simply theorems of the theory expressed in terms of strings. Thus the two representations look substantially the same with respect to "high level" theorems.

3) Ill-formed formulas can be mentioned. This is of course impossible in an axiomatization in terms of the abstract syntax.

The properties of wffs relevant to our theory have been defined by the predicates FR, FRN, GEB and SBT. FR(x,f) is true iff the variable x has at least one free occurrence in the wff f, while FRN(x,n,f) and GEB(x,n,f) are respectively true when the variable x occurs free or bound at the place n in the formula f. These predicates are defined in appendix 1.6. In addition, some generalized selector functions are defined, which evaluate the first or the k-th free occurrence of a variable in a wff, or the number of its free occurrences. The predicate SBT is then defined. It axiomatizes the notion of substitution of a term for any free occurrence of a variable in a wff.

```
∀x t f1 f2.(SBT(x,t,f1,f2) ≡
        ∀n1 n2.((n2=(numbfreeocc(x,n1,f1)*(len(t)-1))+n1) ⊃
            ((¬INDVAR(n1 gl f1) ⊃ (n1 gl f1)=(n2 gl f2)) ∧
             (INDVAR(n1 gl f1) ⊃ ((FRN(x,n1,f1)⊃SUBT(t,f2,n2)) ∧
                                 (¬FRN(x,n1,f1) ⊃ INVART(n1,f1,n2,f2)))) )))

∀t f2 n2.(SUBT(t,f2,n2) ≡ ∀x2 k.((k gl t)=x2 ⊃ FRN(x2,n2-(len(t)-k),f2)))),

∀n f1 n1 f2.(INVART(n,f1,n1,f2) ≡ ((GEB(n1 gl f2,n1,f2) ≡ GEB(n gl f1,n,f1)) ∧
                        (FRN(n1 gl f2,n1,f2) ≡ FRN(n gl f1,n,f1)) ∧ (n1 gl f2)=(n gl f1)))
```

In the previous definition, n1 is any position in the string f1 and n2 is the corresponding position in f2. The auxiliary predicate SUBT states that the variables appearing in the term t substituted for a free occurrence of the variable x are still free. INVART defines which properties of f1 are still true for f2. If the term t is a variable, then SBT reduces to SBV.

```
∀x1 x2 f1 f2.(SBV(x1,x1,f1,f2) ≡
        ∀n.((¬INDVAR(n gl f1) ⊃ (n gl f1)=(n gl f2) ∧
            (INDVAR(n gl f1) ⊃ ((FRN(x1,n,f1) ⊃ FRN(x2,n,f2)) ∧
                               (¬FRN(x1,n,f1) ⊃ INVARV(n,f1,f2)))))),

∀n f1 f2.(INVARV(n,f1,f2) ≡ ((GEB(n gl f2,n,f2) ≡ GEB(n gl f1,n,f1)) ∧
                (FRN(n gl f2,n,f2) ≡ FRN(n gl f1,n,f1)) ∧ (n gl f2)=(n gl f1))),
```

The proof of the equivalence of SBT and SBV when t is a variable is very simple. It is based on the fact that n2 coincides with n1 when the term t has length 1 (see appendix 4). The function sbt (sbv) evaluates to the string representing the result of substituting a term (variable) for every free occurrence of a variable in a given wff. sbt and sbv are defined from the predicates SBT and SBV as follows:

```
∀x t f1 f2.(SBT(x,t,f1,f2) ≡ sbt(x,t,f1)=f2)

∀x1 x2 f1 f2.(SBV(x1,x2,f1,f2) ≡ sbv(x1,x2,f1)=f2)
```

The problem of finding the best way of defining functions in FOL is crucial: in the axiom system given in this paper a uniform way has not been followed. In defining the substitution we are interested in properties of the functions sbt and sbv and in drawing conclusions from the fact that a substitution has been made. It is thus useful to have a predicate which defines the relation between formulas before and after a substitution instead of infering it from the definitions of the functions (stated for example as a system of equations, as in Kleene 1952). One of the motivations of the present experiment was to explore different ways of defining functions. We do not yet have enough examples of proofs to make a clear statement about this matter.

2.2.2 Rules of inference, deductions and the notion of provability

The rules of inference are defined by the predicates in appendix 1.7. The rules with one premise, are expressed by means of a binary predicate whose arguments are two sequences of wffs (**sq, pf**) which satisfy PROOFTREE. The predicate is true iff pf is the scdr of sq and the first element of sq is a wff obtained by applying that particular deduction rule to the first wff of pf. The rules with more antecedents are defined in a similar way.

Derivations are recursively defined as sequences of wffs which either are a single wff or are obtained from one or more derivations by applying one of the deduction rules. The recursion is implicitly stated by saying that there exist objects of sort PROOFTREE which satisfy one of the predicates defining the rules of inference. These sequences represent the linearization of a deduction-tree and are defined as follows:

```
∀sq.(PROOFTREE(sq) ≡
      (FORM(sq) v
      ∃pf.(ORI(sq,pf) v ANDE(sq,pf) v FALSEE(sq,pf) v NOTI(sq,pf) v NOTE(sq,pf) v IMPLI(sq,pf)) v
      ∃pf x t.(GENI(sq,pf,x,t) v GENE(sq,pf,x,t) v EXI(sq,pf,x,t)) v
      ∃pf1 pf2.(ANDI(sq,pf1,pf2) v FALSEI(sq,pf1,pf2) v IMPLE(sq,pf1,pf2)) v
      ∃pf1 pf2 x1 x2.EXE(sq,pf1,pf2,x1,x2) v
      ∃pf1 pf2 pf3.ORE(sq,pf1,pf2,pf3) )) ;;
```

A sequence of wffs is a prooftree if either it consists of a single wff or one of the following alternatives holds: there exists another prooftree and a one premise deduction rule has been applied; there exist two prooftrees and one of the two premises rules has been applied; finally, there are three prooftrees and the predicate defining the v-elimination rule is true. Note that the root of a prooftree is not necessarily a theorem in a given theory. A predicate **DEPEND** has been defined which is true if a given wff is a dependence for the root of a prooftree. The axioms about **DEPEND** allows to decide all the dependencies of a prooftree.

Since some of the deduction rules (the implication introduction, for instance) eliminate dependencies, not all the leaves of a prooftree pf are dependencies for a wff f such that f=**scar**(pf). The predicate **DEPEND** is true only for those leaves of the prooftree which the formula f actually depends on. Its definition is shown in appendix 1.8. The axioms **DEPEND** state which dependencies do not change by applying the deduction rules and are transferred from one prooftree to the other. The axioms **NDEPND** state which rules discharge dependencies in a given prooftree.

Using this notion of dependence ,the provability of a formula in a theory is defined as follows:

```
∀f.(BEW(f) ≡ ∃sq.(PROOFTREE(sq) ∧ f=scar(sq) ∧ ∀f1.(DEPEND(sq,f1) ⊃ AXIOM(f1))));;
```

A wff f is a theorem in a given theory if there exists a prooftree whose first element is f and whose only dependencies are axioms in that theory. We have limited our attention to theories in which axioms have no free variables. This property is defined by the axiom:

```
∀x f.(AXIOM(f) ⊃ ¬FR(x,f));;
```

Section 2.3 The main proof in the many sorted logic

The main theorem we have proved in this axiomatization of the metamathematics states that if ∀x y.wff is provable in some theory, then ∀y x.wff is also provable. We have chosen this theorem because, even if very simple, it involves basic notions of provability, substitution and universal quantification. Its proof is found in appendices 5.1-2. The theorem depends on the first three lines of the proof. The first step is a lemma stating that ∀x wff.sbl(x,x,f)=wff, i.e. substituting a variable x for any free occurrence of x in wff doesn't change that wff. Steps two and three give simple facts about sequences. The theorem is then proved by instantiating two other lemmas: 1) if ∀x.wff is a theorem, then wff is also a theorem; 2) if wff is provable, then x cannot be free in the dependencies of the proof of wff and so ∀x.wff is provable. This is of course true only for theories with no free variables in their axioms.

The only property of the inference rules used in this proof involves universal quantification. The restriction on the applicability of the ∀-introduction rule is that the variable to be universally quantified in a wff must not appear free in any of its dependencies. This restriction is reflected in our axiomatization by the predicate APGENI. In this proof APGENI is satisfied because if wff is provable, its dependencies are axioms with no free variables.

The following is an informal proof of the above theorems. If ∀x.wff is provable, then there is a prooftree pf whose first string is ∀x.wff. The sequence (∀x.wff) cc pf is still a prooftree. It is obtained by applying the ∀-elimination rule. The application of this rule doesn't add any dependency to the prooftree. As its only dependencies are axioms, it follows from the definition of BEW that wff is a theorem. On the other hand, if wff is a theorem there exists a prooftree pf whose first element is wff. By applying the ∀-introduction rule to pf we obtain the prooftree (∀x.wff) cc pf. This rule is applicable since theorems have no free variables in their dependencies. It follows that ∀x.wff is a theorem. If ∀x y.wff is provable then ∀x.wff and wff are provable using the first lemma. Finally, we can quantify first over x and then over y, obtaining ∀y x.wff as a theorem.

Section 2.4 Another axiomatization

A different axiomatization has been given in an earlier version of FOL where there was no facility for creating sorts. We present it here as we want to do some comparisons between proofs, and discuss some of the features of FOL. Some differences between the two axiomatizations are due to the new features available in FOL. They will be discussed in the next section. Here we only discuss the difference between the definition of formulas and terms. The list of all the axioms can be found in appendices 2.1-8.

In this axiomatization, formulas and terms are still represented as the string of the symbols appearing in them. They are defined as strings that can be decomposed into a sequence of substrings recording the construction of that formula or term from elementary formulas and individual variables, according to the usual formation rules (see appendix 2.5 for the list of axioms). These sequences are defined by the predicate TERMSEQ for terms and FRR for wffs. A sequence satisfies the predicate TERMSEQ if it represents the history of the construction of its first element (the term to be defined), starting from symbols, functions and individual variables. Similarly, a string is a wff if there exists a sequence which satisfies the predicate FRR and represents the history of the construction of that wff from elementary formulas and the logical connectives.

SECTION 3 THE PROOFS

In this section we look at the proofs appearing in the appendices, in order to explore the features of FOL that need improving and their use in carrying out formal proofs.

Section 3.1 A look at sorts

As already noted, the primary difference between the two axiomatizations we presented is the introduction of a many sorted logic. In the earlier version of FOL there was no facility for creating sorts, but it soon became evident that relativization of wffs to predicates was desirable. The notion of partially ordered sorts was a natural outgrowth. The axioms in the sorted logic are simpler and more readable and, most important, proofs are considerably shorter. First of all, in the axiomatization done in the earlier version of FOL the partial order of sorts wasn't explicit and was to be derived as a theorem. In the proofs shown in the appendices these theorems appear as dependencies. At the moment FOL has no facility for using already proved statements as lemmas in making new proofs. In FOL there is also the possibility of declaring for each function symbol the sorts of its arguments and of its value. These sorts were defined in the original version by additional axioms. For example, together with the definition of the functions **sbt** and **sbv**, the second axiomatization has two extra axioms.

∀x t f1.((INDVAR(x) ∧ TERM(t) ∧ FORM(f1)) ⊃ FORM(sbt(x,t,f1)));;

∀x1 x2 f1.((INDVAR(x1) ∧ INDVAR(x2) ∧ FORM(f1)) ⊃ FORM(sbv(x1,x2,f1)));;

Proofs are shorter in a many sorted logic. As an example, we can examine the two proofs in appendices 5.1-2 and 5.5-6. The second proof is longer because the explicit assumption that **x** is an individual variable and **f** is a wff must be made, and the symbol **⊃** must be introduced at the end of the proof, to discharge this assumption. Note that in this proof the statements labeled **TH2** and **TH3** appear as dependencies and the proof would have been even longer if we had proved them there. Another difference between the two proofs is that, in the second one, we had to use the axiom previously mentioned stating that the result of substituting a term t for every free occurrence of a variable in a wff is still a wff. The different axiomatization of wffs and terms only influences the length of the proofs in appendix 3.1,-2,-3. All the other proofs are shorter only due to the presence of sorts in FOL. Furthermore, note that proofs in the second axiomatization have more dependencies since all the theorems about the partial order of sorts have been assumed.

Section 3.2 The unify and tautology commands

FOL proofs are greatly simplified by the existence of the commands **TAUT** and **TAUTEQ**. They decide if a given formula is a tautological consequence of a specified set of wffs. The difference between **TAUT** and **TAUTEQ** is that the latter uses properties of the equality and the former doesn't. These commands make proofs shorter since they allow to decide every propositional sentence in one step. As a consequence, the rules of inference most frequently used manipulate quantifiers. The form of almost all the proofs we presented is the same. First of all, the right instantiations of the relevant axioms and theorems are done. Then the propositional consequences are asserted by using **TAUT** and **TAUTEQ**. The tautology commands cannot of course manipulate the quantifiers appearing in

statements. Hence, the statements produced by them have quantifiers as main symbols or it is necessary to introduce a quantifier to proceed in the proof. After the right introductions or eliminations have been done to them, the tautology commands are used again. This process is iterated until the completion of the proof.

The command UNIFY decides if a given wff can be obtained by instantiation of quantified variables or introduction of them for free occurrences of variables or terms in a second wff. The code for this command has been written by Ashok Chandra and is still in an experimental stage. In the proofs presented here, this command has been essentially used for the simultaneous introduction of the existential quantifier. As an example, consider the following assumption:

```
1  ∀x.(P(x)⊃(Q(f1 c f2)∧∀l.R(l)))  (1)  ASSUME
```

the command

```
unify ∃x.(P(x)⊃∃f.(Q(f)∧R(g(l)))),1;
```

deduces in a single step

```
2  ∃x.(P(x)⊃∃f.(Q(f)∧R(g(l)))) (6) UNIFY 1
```

A good example of a combined use of these features is found in appendix 3.3:

```
19 FRR((x1 gen f) cc SQ) (SEQUENCE((x1 gen f) cc SQ)∧(((x1 gen f) cc U)≠
   SLAMBDA∧(ELF(scar((x1 gen f) cc SQ))∨(FRR(scdr((x1 gen f) cc SQ))∧
   ∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar((s1 gen f) cc SQ)=NEG(s1)∧
   find(1,s1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 dis s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 con s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 impl s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 gen s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(s1 ex s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ)))))))))))))))) --- ∀E WFF1 (x1 gen f) cc SQ

20 STRING(x1)∧(STRING(f)∧((scar((x1 gen f) cc SQ)=NEG(x1)∧
   find(1,x1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 dis f)∧
   find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 con f)∧
   find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 impl f)∧
   find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 gen f)∧(INDVAR(x1)∧
   find(1,f,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(x1 ex f)∧(INDVAR(x1)∧
   find(1,f,scdr((x1 gen f) cc SQ))))))))))) (1 2 3 4 5 6 7 8 11) --- TAUTEQ 1:19

21 ∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar((s1 gen f) cc SQ)=NEG(s1)∧
   find(1,s1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 dis s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 con s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 impl s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 gen s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(s1 ex s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ)))))))))))))))) (1 2 3 4 5 6 7 8 11) --- UNIFY 20
```

Line 19 is the instantiation of an axiom. Line 20 is generated by the command,

```
TAUTEQ 19:#2#2#2#2#2#1#1[s1←f : s2←x1] 1:19;
```

note how the use of the FOL subpart designators allows us to mention the desired subpart of 19, without having to retype it. In addition we can do the appropriate substitutions. Line 21 is just a use of UNIFY:

```
UNIFY 19:•2•2•2•2•2 20;
```

Because we can mention the conclusion, without writing it down explicitly, the amount of typing necessary is severely reduced. Without UNIFY, line 21 would have required two ∃-introductions and the commands would have been:

```
∃I 20 x1←s1 OCC 1,2,3,4,7,8,11,12,15,16,19,20,23,24;
```

```
∃I 20 f ←s2 OCC 1,5,9,13,17,18,21,22;
```

We do not enter into a detailed discussion of the command UNIFY. It is our intension to do it elsewhere. It should be thought of as the routine which handles quantifiers in "simple" inferences. As seen above, the saving to a user can be large.

SECTION 4 CONCLUSION

The desire to represent mathematics in a computer in a feasible way certainly requires the facility to discuss metamathematical notions. The axiomatization presented here only treats the syntactic part of the problem. Any mention of the models involved needs the addition of set theory to the axiomatization. However, it is clear from the simple theorems we proved that any practical system needs more extensive features even to do a satisfactory job of writing down the theorems we might want.

An important point for future work is how (in a practical way) to use these theorems. Consider for instance:

∀x1 x2 f.(BEW(x1 gen (x2 gen f)) ⊃ BEW(x2 gen (x1 gen f)))

What we mean by reflection principle is a rule of FOL which says:

```
   //BEW(f)          //in meta FOL
  /--------
    ///f            / in FOL
```

That is, if in the axiomatization of the metamathematics of FOL, we can prove the existence of an FOL proof of f, then we can assert f in FOL. Suppose we have a proof in FOL of ∀x y.wff. Then instantiating the above theorem gives us

BEW(x gen (y gen wff)) ⊃ BEW(y gen (x gen wff))

Since we started with a proof of ∀x y.wff in FOL and BEW represents the proof predicate for FOL, we can conclude BEW(x gen (y gen wff)). Using modus ponens we get BEW(y gen (x gen wff)), and using the above rule we can conclude ∀y x.wff in FOL.

The exact form of such a rule requires more examples of proofs and is one of the main reasons for doing the example in the memo. It is not just a proof checking exercise, but a case study for fundamental questions of representing mathematical information in a computer. Using metamathematics also prepares the way for more comprehensive systems which can formally discuss how they reason. That is exactly what the metamathematics is good for.

APPENDIX 1

THE AXIOMS IN THE MANY SORTED LOGIC

1.1 Natural numbers

AXIOM NUMB:

$\forall n1\ n2\ n3.\quad (n1=n2 \supset (n1=n3 \supset n2=n3)),$

$\forall n1\ n2.\quad (n1=n2 \supset \mathrm{succ}(n1)=\mathrm{succ}(n2)),$

$\forall n1.\quad 0 \neq \mathrm{succ}(n1),$

$\forall n1\ n2\quad (\mathrm{succ}(n1)=\mathrm{succ}(n2) \supset n1=n2),$

$\forall n1.\quad n1+0=n1,$

$\forall n1\ n2.\quad n1+\mathrm{succ}(n2)=\mathrm{succ}(n1+n2),$

$\forall n1.\quad n1*0=0,$

$\forall n1\ n2.\quad n1*\mathrm{succ}(n2)=(n1*n2)+n1\ ;;$

AXIOM INDCT:

$(F(0) \land \forall n.(F(n) \supset F(n+1))) \supset \forall n.F(n)\ ;;$

AXIOM DEFN:

$\forall n.\quad (\mathrm{succ}(n)-1)=n,$

$\forall n1\ n2.\quad \mathrm{succ}(n1)-n2=n1-(n2-1),$

$\forall n1\ n2\ n3.\quad (n1<n2 \equiv \exists n3.(n3 \neq 0 \land n1+n3=n2)),$

$\forall n1\ n2.\quad (n1 \leq n2 \equiv (n1<n2) \lor (n1=n2)),$

$\forall n1\ n2.\quad (n2>n1 \equiv n1<n2),$

$\forall n1\ n2.\quad (n2 \geq n1 \equiv n1 \leq n2)\ ;;$

1.2 The set of symbols

AXIOM SYM:

$\forall a.\ (\mathrm{SYM}(a) \equiv a=\mathrm{LPARSYM} \lor a=\mathrm{RPARSYM} \lor a=\mathrm{ORSYM} \lor a=\mathrm{ANDSYM} \lor a=\mathrm{IMPSYM} \lor$
$a=\mathrm{FALSESYM} \lor a=\mathrm{FORALLSYM} \lor a=\mathrm{EXISTSYM})\ ;;$

1.3 Strings

AXIOM STRING:

$\forall s.\quad s=\mathrm{car}(s)\ c\ \mathrm{cdr}(s),$

$\forall s1\ s2.\quad (s1=\mathrm{LAMBDA} \supset \mathrm{car}(s1\ c\ s2)=\mathrm{car}(s2)),$

$\forall s1\ s2.\quad (s1 \neq \mathrm{LAMBDA} \supset \mathrm{car}(s1\ c\ s2)=\mathrm{car}(s1)),$

$\forall s1\ s2.\quad (s1=\mathrm{LAMBDA} \supset \mathrm{cdr}(s1\ c\ s2)=\mathrm{cdr}(s2)),$

$\forall s1\ s2.\quad (s1 \neq \mathrm{LAMBDA} \supset \mathrm{cdr}(s1\ c\ s2)=\mathrm{cdr}(s1)),$

$\forall s.\quad (s\ c\ \mathrm{LAMBDA}=\mathrm{LAMBDA}\ c\ s),$

$\forall s.\quad s\ c\ \mathrm{LAMBDA}=s,$

$\forall s1\ s2\ s3.\quad (s1\ c\ (s2\ c\ s3)=(s1\ c\ s2)\ c\ s3),$

$\forall a.\quad (\mathrm{len}(a)=1 \lor a=\mathrm{LAMBDA}),$

$\forall s.\quad \mathrm{len}(s) \geq 0,$

$\forall s1\ s2.\quad \mathrm{len}(s1\ c\ s2)=\mathrm{len}(s1)+\mathrm{len}(s2),$

$\forall s.\quad (\mathrm{len}(s)=1 \supset \mathrm{ATOM}(s)),$

$\forall s.\quad 0\ \mathrm{gl}\ s \equiv \mathrm{LAMBDA},$

```
     ∀s.          1 gl s=car(s) ,
     ∀s n.        ((n>1)⊃((n gl s)=((n-1) gl cdr(s)))) ;;
```

AXIOM SUBSTRDEF:

```
     ∀n1 n2 s1 s2      (SUBSTP(s1,s2,n1,n2) ≡ (len(s2)=n2-n1+1∧(∀n.(n≥n1∧n≤n2 ⊃
                             n gl s1=(n-n1+1) gl s2)))),
     ∀n1 n2 s1 s2      (SUBSTP(s1,s2,n1,n2) ≡ substring(s1,n1,n2)=s2) ,
     ∀s1 s2.           (SUBS(s1,s2) ≡ ∃n1 n2.SUBSTP(s1,s2,n1,n2));;
```

The value of substring(s1,n1,n2) is the substring of s1 whose first element is the n1th element of s1 and whose last element is the n2th element.

AXIOM DISEQ:

```
     ∀g1 g2. (¬(g1=g2) ≡ g1≠g2) ;;
```

AXIOM EQS:

```
     ∀s1 s2. (∀n.(n gl s1=n gl s2) ≡ s1=s2) ;;
```

AXIOM COMP:

```
     ∀f.          e(f)=(LPARSYM c f) c RPARSYM ,
     ∀f1 f2.      f1 dis f2=(e(f1) c ORSYM) c e(f2) ,
     ∀f1 f2.      f1 impl f2=(e(f1) c IMPSYM) c e(f2) ,
     ∀f.          neg(f)=(f impl FALSESYM) ,
     ∀f1 f2.      f1 con f2=(e(f1) c ANDSYM) c e(f2) ,
     ∀x f2.       x gen f2=(FORALLSYM c x) c f2 ,
     ∀x f2.       x ex f2=(EXISTSYM c x) c f2 ;;
```

1.4 Formulas

AXIOM TERM:

```
                  TERMSEQ (0,LAMBDA) ,
     ∀n s.        (TERMSEQ(n,s) ≡ (∃n1.(TERM(substring(s,1,n1)) ∧
                       TERMSEQ(n-1,substring(s,n1+1,len(s)))))) ,
     ∀s.          (TERM(s) ≡ INDVAR(s) ∨ ∃n fn.(fn=car(s) ∧ n=arity(fn) ∧ TERMSEQ(n,cdr(s)))) ;;
```

AXIOM WFF:

```
     ∀s.          (ELF(s) ≡ (s=FALSESYM ∨ PREDPAR0(s) ∨ ∃n P.(P=car(s) ∧ n=arity(P) ∧
                       TERMSEQ(n,cdr(s))))),
     ∀s.          (FORM(s) ≡ (ELF(s) ∨
                       ∃x f.((s=x gen f) ∨ (s=x ex f)) ∨
                       ∃f1 f2.((s = f1 dis f2) ∨ (s = f1 con f2) ∨ (s = f1 impl f2)) ∨
                       ∃f.s=neg (f))) ;;
```

1.5 Sequences

AXIOM SEQ:

```
     ∀sq          sq=scar(sq) cc scdr(sq) ,
     ∀sq1 sq2.    (sq1=SLAMBDA ⊃ scar(sq1 cc sq2)=scar(sq2)) ,
     ∀sq1 sq2.    (sq1≠SLAMBDA ⊃ scar(sq1 cc sq2)=scar(sq1)) ,
     ∀sq1 sq2.    (sq1=SLAMBDA ⊃ scdr(sq1 cc sq2)=scdr(sq2)) ,
```

```
     ∀sq1 sq2.        (sq1≠SLAMBDA ⊃ scdr(sq1 cc sq2)=scdr(sq1) cc sq2) ,
     ∀sq.             sq cc SLAMBDA=SLAMBDA cc sq ,
     ∀sq.             sq cc SLAMBDA=sq ,
     ∀sq1 sq2 sq3.    (sq1 cc (sq2 cc sq3)=(sq1 cc sq2) cc sq3) ,
     ∀s.              (slen(s)=1∨s=SLAMBDA) ,
     ∀sq.             slen(sq)≥0 ,
     ∀sq1 sq2.        slen(sq1 cc sq2)=slen(sq1)+slen(sq2) ,
     ∀sq.             0 sgl sq=SLAMBDA ,
     ∀sq.             1 sgl sq=scar(sq) ,
     ∀n sq.           ((n>1) ⊃ ((n sgl sq)=((n-1) sgl scdr(sq)))) ;;

AXIOM SUBSEQDEF:
     ∀n1 n2 sq1 sq2.   (SUBSEP(sq1,sq2,n1,n2) ≡ (slen(sq2)=n2-n1+1 ∧
                      (∀n.(n≥n1∧n≤n2 ⊃ n sgl sq2=(n-n1+1) sgl sq1)))) ,
     ∀n1 n2 sq1 sq2.   (SUBSEP(sq1,sq2,n1,n2) ≡ subseq(sq1,n1,n2)=sq2) ,
     ∀sq1 sq2.        (SUBSSE(sq1,sq2) ≡ ∃n1 n2.(SUBSEP(sq1,sq2,n1,n2))) ;;

AXIOM EQSQ:
     ∀sq1 sq2.    (∀n.(n sgl sq1=n sgl sq2) ⊃ sq1=sq2) ;;
```

1.6 Free and bound variables and the substitution

```
AXIOM BOUNDV:
     ∀x n f.      (GEB(x,n,f) ≡ ∃s1 s2 f1.(len(s1)+1<n ∧ n<(len(f)-len(s2)) ∧
                       (x=n gl f)∧((f=(s1 c ((x gen f1) c s2)))∨(f=(s1 c ((x ex  f1) c s2)))))) ;;

AXIOM FREEV:
     ∀x n f.      (FRN(x,n,f) ≡ (x=(n gl f) ∧ ¬GEB(x,n,f))) ,
     ∀x f.        (FR(x,f) ≡ ∃n.FRN(x,n,f));;

AXIOM FIRSTFRDF:
     ∀x n f.      (FIRSTFREE(x,n,f) ≡ (FRN(x,n,f) ∧ ∀n1.(x=n1 gl f ⊃ (n1≥n ∨ GEB(x,n1,f))))) ,
     ∀x n f.      (FIRSTFREE(x,n,f) ≡ firstfreeocc(x,f)=n) ;;

AXIOM  KFREEOCCDF:
     ∀x k n f.    (KTHFREEOCC(x,k,n,f) ≡ ((k=0 ∧ n=1) ∨
                       (n=len(f) ∧ ∀n2.(n2>kthfreeocc(x,k-1,f) ⊃ ¬FRN(x,n2,f))) ∨
                       (FRN(x,n,f) ∧ ∀n1.((n1<k ∧ n1>0) ⊃ ∃n2.(n2<n ∧ KTHFREEOCC(x,n1,n2,f)))))),
     ∀x k n f.    (KTHFREEOCC(x,k,n,f) ≡ kthfreeocc(x,k,f)=n) ,
     ∀x k n f.    (KTHFREEOCC(x,k,n,f) ⊃ numbfreeocc(x,n,f)=k),
     ∀x k n f.    (numbfreeocc(x,n,f)=k ⊃ (KTHFREEOCC(x,k,n,f) ∨
                       (n<kthfreeocc(x,k,f) ∧ n>kthfreeocc(x,k-1,f))));;
AXIOM SUBSTDF:
     ∀x t f1 f2.   (SBT(x,t,f1,f2) ≡ ∀n1 n2.((n2=(numbfreeocc(x,n1,f1)*(len(t)-1))+n1)⊃
                       ((¬INDVAR(n1 gl f1) ⊃ n1 gl f1 = n2 gl f2)∧
                       (INDVAR(n1 gl f1) ⊃ ((FRN(x,n1,f1) ⊃ SUBT(t,f2,n2))∧
                                  (¬FRN(x,n1,f1)⊃INVART(n1,f1,n2,f2))))))),
     ∀t f2 n2.     (SUBT(t,f2,n2) ≡ ∀x2 k.(((k gl t) = x2) ⊃ FRN(x2,n2-(len(t)-k),f2))),
     ∀n f1 n1 f2.  (INVART(n,f1,n1,f2) ≡ ((GEB(n1 gl f2,n1,f2) ≡ GEB(n gl f1,n,f1))∧
                       (FRN(n1 gl f2,n1,f2) ≡ FRN(n gl f1,n,f1)) ∧ n1 gl f2=n gl f1))
     ∀x t f1 f2.   (SBT(x,t,f1,f2) ≡ sbt(x,t,f1)=f2) ;;
```

```
AXIOM SUBDEF:
    ∀x1 x2 f1 f2.(SBV(x1,x2,f1,f2) ≡ ∀n.((¬INDVAR(n gl f1) ⊃ n gl f1 = n gl f2)∧
                    (INDVAR(n gl f1) ⊃ ((FRN(x1,n,f1)⊃FRN(x2,n,f2)) ∧
                        (¬FRN(x1,n,f1)⊃INVARV(n,f1,f2)))))),
    ∀n f1 f2.     (INVARV(n,f1,f2) ≡ ((GEB(n gl f2,n,f2) ≡ GEB(n gl f1,n,f1))∧
                    (FRN(n gl f2,n,f2) ≡ FRN(n gl f1,n,f1)) ∧ n gl f2 = n gl f1)),
    ∀x x1 f1 f2. (SBV(x,x1,f1,f2) ≡ sbv(x,x1,f1)=f2);;
```

1.7 Rules of inference

```
AXIOM ANDIRUL:
    ∀sq pf1 pf2. (ANDI(sq,pf1,pf2) ≡ ∃f1 f2.(scdr(sq)=(pf1 cc pf2) ∧ scar(sq)=f1 con f2 ∧
                    f1=scar(pf1) ∧ f2=scar(pf2))),
    ∀sq pf.       (ANDE(sq,pf) ≡ ∃f1 f2.(scdr(sq)=pf ∧ scar(sq)=f1 ∧ ((f1 con f2)=scar(pf))∨
                    (f2 con f1)=scar(pf))));;

AXIOM FALSERUL:
    ∀sq pf1 pf2. (FALSEI(sq,pf1,pf2) ≡ ∃f1.((scdr(sq)=(pf1 cc pf2))∧
                    (scar(sq)=FALSESYM) ∧ (neg(f1)=scar(pf1)) ∧ (f1=scar(pf2)))),
    ∀sq pf.       (FALSEE(sq,pf) ≡ ∃f.((scar(pf)=FALSESYM) ∧ f=scar(sq) ∧ scdr(sq)=pf));;

AXIOM IMPLRUL:
    ∀sq pf1 pf2. (IMPLE(sq,pf1,pf2) ≡ ∃f1 f2.((scdr(sq)=(pf1 cc pf2))∧
                    (scar(pf1)=(f1 impl f2)) ∧ (scar(sq)=f2 ∧ (scar(pf2) = f1))),
    ∀sq pf f1.    (IMPLID(sq,pf,f1) ≡ (scdr(sq)=pf ∧ ∃f2.((scar(sq)=(f1 impl f2))∧
                    (f2=scar(pf)) ∧ ∃n.(f1 = (n sgl pf))))),
    ∀sq pf.       (IMPLI(sq,pf) ≡ ∃f.IMPLID(sq,pf,f1);;

AXIOM NEGRUL:
    ∀sq pf f.     (NOTID(sq,pf,f) ≡ (scdr(sq)=pf ∧ scar(sq)=f ∧ (scar(pf)=FALSESYM) ∧
                    ∃n.(n sgl pf)=f)) ,
    ∀sq pf.       (NOTI(sq,pf) ≡ ∃f.NOTID(sq,pf,f),
    ∀sq pf f.     (NOTED(sq,pf,f) ≡ (scdr(sq)=pf ∧ (scar(pf) = FALSESYM)∧
                    ∃n.((n sgl pf)=f) ∧ (f=neg(scar(sq))))),
    ∀sq pf.       (NOTE(sq,pf) ≡ ∃f.NOTED(sq,pf,f);;

AXIOM ORRUL:
    ∀sq pf.                 (ORI(sq,pf) ≡ (scdr(sq)=pf ∧ ∃f1 f2.((scar(sq)=(f1 dis f2)) ∧
                              f1=scar(pf)) ∨ (f2=scar(pf))))),
    ∀sq pf1 pf2 pf3 f1 f2.  (ORED(sq,pf1,pf2,pf3,f1,f2) ≡ (scdr(sq)=(pf1 cc (pf2 cc pf3)) ∧
                              (scar(pf1)=(f1 dis f2) ∧ ∃f3.(scar(pf2)=f3) ∧ scar(sq)=f3 ∧
                              (scar(pf3)=f3)) ∧ ∃n1.(n1 sgl pf2)=f1)∧∃n1.(n1 sgl pf3)=f2))),
    ∀sq pf1 pf2 pf3.        (ORE(sq,pf1,pf2,pf3) ≡ ∃f1 f2.ORED(sq,pf1,pf2,pf3,f1,f2));;

AXIOM EXRUL:
    ∀sq pf x t.             EXI(sq,pf,x,t) ≡ ∃f1.((scdr(sq)=pf1) ∧ (scar(sq)=(x ex f1)) ∧
                              scar(pf)=sbt(x,t,f1))) ,
    ∀sq pf1 pf2 x1 x2 f1.   (EXED(sq,pf1,pf2,x1,x2,f1) ≡ ((scdr(sq)=(pf1 cc pf2)) ∧
                              (scar(pf1)=(x1 ex f1)) ∧ (scar(sq)=scar(pf2)) ∧
                              ∃n.((n sgl pf2)=sbt(x1,x2,f1) ∧ EXAPPL(x2,pf2,f1)))),
    ∀sq pf1 pf2 x1 x2.      (EXE(sq,pf1,pf2,x1,x2) ≡ ∃f1.EXED(sq,pf1,pf2,x1,x2,f1),
    ∀x pf f.                (EXAPPL(x,pf,f) ≡ (¬FR(x,scar(pf)) ∧ ¬FR(x,f) ∧ ∀f1.(DEPEND(pf,f1) ⊃
                              ¬FR(x,f1))));;
```

```
AXIOM GENRUL:
      ∀sq sq1 x t.          (GENE(sq,sq1,x,t) ≡ (scdr(sq)=sq1 ∧ PROOFTREE(sq1) ∧
                              ∃f.(scar(sq1)=x gen f ∧ scar(sq) = sbt(x,t,f)))),
      ∀sq sq1 x1 x2.        (GENI(sq,sq1,x1,x2) ≡ (scdr(sq)=sq1 ∧ PROOFTREE(sq1) ∧
                              ∃f.(scar(sq)=x1 gen f ∧ scar(sq1) = sbt(x1,x2,f) ∧ APGENI(x2,sq1)))),
      ∀x sq.                (APGENI(x,sq) ≡ (∀f.(DEPEND(sq,f) ⊃ ¬FR(x,f))) ∧ PROOFTREE(sq)),
      ∀pf.∃x.                APGENI(x,pf);;
```

1.8 Deduction

```
AXIOM PROOF:
      ∀sq.          (PROOFTREE(sq) ≡ (FORM(sq) ∨
                        ∃pf.(ORI(sq,pf) ∨ ANDE(sq,pf) ∨ FALSEE(sq,pf) ∨ NOTI(sq,pf) ∨ NOTE(sq,pf) ∨
                              IMPLI(sq,pf)) ∨
                        ∃pf x f.(GENI(sq,pf,x,f) ∨ GENE(sq,pf,x,f) ∨ EXI(sq,pf,x,f)) ∨
                        ∃pf1 pf2.(ANDI(sq,pf1,pf2) ∨ FALSEI(sq,pf1,pf2) ∨ IMPLE(sq,pf1,pf2)) ∨
                        ∃pf1 pf2 x1 x2. EXE(sq,pf1,pf2,x1,x2) ∨
                        ∃pf1 pf2 pf2 pf3.ORE(sq,pf1,pf2,pf3)));;

AXIOM DEPNDG:
      ∀sq f.        (DEPEND(sq,f) ⊃ SUBSSE(f,sq)) ,
      ∀sq f.        (f=sq ⊃ DEPEND(sq,f)) ;;

AXIOM DEPEND:
      ∀pf pf1 f.          ((pf1=scdr(pf) ⊃ (DEPEND(pf,f) ≡  DEPEND(pf1,f))) ≡
                              (ORI(pf,pf1) ∨ ANDE(pf,pf1) ∨ FALSEE(pf,pf1) ∨
                              ∃f1.((NOTID(pf,pf1,f1) ∨ NOTED(pf,pf1,f1) ∨ IMPLID(pf,pf1,f1)) ∧ f1≠f) ∨
                              ∃x f.(GENI(pf,pf1,x,t) ∨ GENE(pf,pf1,x,t) ∨ EXI(pf,pf1,x,t)))) ,
      ∀pf pf1 pf2 f.      ((((pf1 cc pf2=scdr(pf)) ∨ (pf2 cc pf1=scdr(pf))) ⊃
                              ((DEPEND(pf,f) ≡ ((DEPEND(pf1,f) ∨ DEPEND(pf2,f)))) ≡
                              (ANDI(pf,pf1,pf2) ∨ FALSEI(pf,pf1,pf2) ∨ IMPLE(pf,pf1,pf2) ∨
                              ∃x1 x2 f1.(EXED(pf,pf1,pf2,x1,x2,f1) ∧ f≠f1))),
      ∀pf pf1 pf2 pf3 f.  (((((pf1 cc (pf2 cc pf3))=scdr(pf)) ∨
                              ((pf1 cc (pf3 cc pf2))=scdr(pf)) ∨
                              ((pf2 cc (pf1 cc pf3))=scdr(pf)) ∨
                              ((pf2 cc (pf3 cc pf1))=scdr(pf)) ∨
                              ((pf3 cc (pf1 cc pf2))=scdr(pf)) ∨
                              ((pf3 cc (pf2 cc pf1))=scdr(pf))) ⊃
                              (DEPEND(pf,f) ≡ (DEPEND(pf1,f) ∨ DEPEND(pf2,f) ∨
                              DEPEND(pf3,f))))≡
                              ∃f1 f2.(ORED(pf,pf1,pf2,pf3,f1,f2) ∧ f≠f1 ∧ f≠f2)) ;;

AXIOM NDEPND:
      ∀pf1 pf2 f.         ((NOTID(pf1,pf2,f) ∨ NOTED(pf1,pf2,f) ∨ IMPLID(pf1,pf2,f)) ⊃
                              ¬DEPEND(pf1,f)),
      ∀pf1 pf2 pf3 x1 x2 f.( EXED(pf1,pf2,pf3,x1,x2,f) ⊃ ¬ DEPEND(pf1,f)) ,
      ∀pf1 pf2 pf3 pf4 f1 f2. (ORED(pf1,pf2,pf3,pf4,f1,f2) ⊃ ¬ DEPEND(pf1,f1) ∧
                              ¬DEPEND(pf1,f2));;
      ∀f.                 (BEW(f) ≡ ∃sq.(PROOFTREE(sq) ∧ f = scar(sq) ∧  ∀f1.(DEPEND(sq,f1) ⊃
                              AXIOM(f1))));;

AXIOM THEORY:
      ∀x f.               (AXIOM(f) ⊃ ¬FR(x,f));;
```

AXIOM INFVAR:

$\forall s.\exists x.\forall n. \quad n\ gl\ s \neq x$;;

APPENDIX 2

THE AXIOMS IN THE LOGIC

2.1 Natural numbers

AXIOM NUMB:

$\forall n1\ n2\ n3.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2) \wedge \mathrm{INTEGER}(n3)) \supset (n1=n2 \supset (n1=n3 \supset n2=n3))),$

$\forall n1\ n2.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset (n1=n2 \supset \mathrm{succ}(n1)=\mathrm{succ}(n2))),$

$\forall n1.\quad (\mathrm{INTEGER}(n1) \supset 0 \neq \mathrm{succ}(n1)),$

$\forall n1\ n2.\quad (\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset (\mathrm{succ}(n1) = \mathrm{succ}(n2) \supset n1 = n2)),$

$\forall n1.\quad (\mathrm{INTEGER}(n1) \supset n1+0=n1),$

$\forall n1\ n2.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset n1 + \mathrm{succ}(n2) = \mathrm{succ}(n1 + n2)),$

$\forall n1.\quad (\mathrm{INTEGER}(n1) \supset n1 * 0=0),$

$\forall n1\ n2.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset n1 * \mathrm{succ}(n2)=(n1 * n2)+n1);;$

AXIOM INDCT:

$(F(0) \wedge \forall x.(\mathrm{INTEGER}(x) \supset (F(x) \supset F(x+1)))) \supset \forall x.(\mathrm{INTEGER}(x) \supset F(x));;$

AXIOM DEFN:

$\forall n.\quad (\mathrm{INTEGER}(n) \supset (\mathrm{succ}(n)-1)=n),$

$\forall n1\ n2.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset \mathrm{succ}(n1)-n2=n1-(n2-1)),$

$\forall n1\ n2\ n3.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2) \wedge \mathrm{INTEGER}(n3)) \supset$
$\quad (n1<n2 \equiv \exists n3.(n3 \neq 0 \wedge n1+n3=n2))),$

$\forall n1\ n2.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset (n1 \leq n2 \equiv (n1<n2) \vee (n1=n2))),$

$\forall n1\ n2.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset (n2>n1 \equiv n1<n2)),$

$\forall n1\ n2.\quad ((\mathrm{INTEGER}(n1) \wedge \mathrm{INTEGER}(n2)) \supset (n2 \geq n1 \equiv n1 \leq n2)),$

2.2 The set of symbols

AXIOM SYM:

$\forall a.\quad (\mathrm{SYM}(a) \equiv a=\mathrm{LPARSYM} \vee a=\mathrm{RPARSYM} \vee a=\mathrm{ORSYM} \vee a=\mathrm{ANDSYM} \vee a=\mathrm{IMPSYM} \vee$
$\quad a=\mathrm{FALSESYM} \vee a=\mathrm{FORALLSYM} \vee a=\mathrm{EXISTSYM})\ ;;$

2.3 Strings

AXIOM STRING:

$\forall s.\quad (\mathrm{STRING}(s) \supset s=\mathrm{car}(s)\ c\ \mathrm{cdr}(s))\ ,$

$\forall s1\ s2.\quad ((\mathrm{STRING}(s1) \wedge \mathrm{STRING}(s2)) \supset (s1=\mathrm{LAMBDA} \supset \mathrm{car}(s1\ c\ s2)=\mathrm{car}(s2))),$

$\forall s1\ s2.\quad ((\mathrm{STRING}(s1) \wedge \mathrm{STRING}(s2)) \supset (s1 \neq \mathrm{LAMBDA} \supset \mathrm{car}(s1\ c\ s2)=\mathrm{car}(s1))),$

$\forall s1\ s2.\quad ((\mathrm{STRING}(s1) \wedge \mathrm{STRING}(s2)) \supset (s1=\mathrm{LAMBDA} \supset \mathrm{cdr}(s1\ c\ s2)=\mathrm{cdr}(s2))),$

$\forall s1\ s2.\quad ((\mathrm{STRING}(s1) \wedge \mathrm{STRING}(s2)) \supset (s1 \neq \mathrm{LAMBDA} \supset \mathrm{cdr}(s1\ c\ s2)=\mathrm{cdr}(s1))),$

$\forall s.\quad ((\mathrm{STRING}(s) \supset (s\ c\ \mathrm{LAMBDA}=\mathrm{LAMBDA}\ c\ s)),$

$\forall s.\quad (\mathrm{STRING}(s) \supset (s\ c\ \mathrm{LAMBDA}=s)),$

$\forall s1\ s2\ s3.\quad ((\mathrm{STRING}(s1) \wedge \mathrm{STRING}(s2) \wedge \mathrm{STRING}(s3)) \supset (s1\ c\ (s2\ c\ s3)=(s1\ c\ s2)\ c\ s3)),$

$\forall s.\quad (\mathrm{STRING}(s) \supset (\mathrm{len}(a)=1 \vee a=\mathrm{LAMBDA})),$

$\forall s.\quad (\mathrm{STRING}(s) \supset \mathrm{len}(s) \geq 0),$

$\forall s1\ s2.\quad ((\mathrm{STRING}(s1) \wedge \mathrm{STRING}(s2)) \supset \mathrm{len}(s1\ c\ s2)=\mathrm{len}(s1)+\mathrm{len}(s2)),$

$\forall s.\quad (\mathrm{STRING}(s) \supset (\mathrm{len}(s)=1 \supset \mathrm{ATOM}(s)),$

```
   ∀s.          (STRING(s) ⊃ 0 gl s=LAMBDA),
   ∀s.          (STRING(s) ⊃ 1 gl s=car(s)),
   ∀s n.        ((STRING(s) ∧ INTEGER(n)) ⊃ ((n>1) ⊃ ((n gl s)=((n-1) gl cdr(s))))),
```

AXIOM SUBSTRDEF:

```
   ∀n1 n2 s1 s2.   ((INTEGER(n1) ∧ INTEGER(n2) ∧ STRING(s1) ∧ STRING(s2)) ⊃
                      (SUBSTP(s1,s2,n1,n2) ≡ (len(s2)=n2-n1+1 ∧ (∀n.(n≥n1 ∧ n≤n2 ⊃
                      n gl s1=(n-n1+1) gl s2))))),
   ∀n1 n2 s1 s2.   ((INTEGER(n1) ∧ INTEGER(n2) ∧ STRING(s1) ∧ STRING(s2)) ⊃
                      (SUBSTP(s1,s2,n1,n2) ≡ substring(s1,n1,n2)=s ,
   ∀s1 s2.         ((STRING(s1) ∧ STRING(s2)) ⊃ (SUBS(s1,s2) ≡ ∃n1 n2.SUBSTP(s1,s2,n1,n2)));;
```

AXIOM DISEQ:

```
   ∀g1 g2.         (¬(g1=g2) ≡ g1≠g2) ;;
```

AXIOM EQS:

```
   ∀s1 s2.         ((STRING(s1) ∧ STRING(s2)) ⊃ (∀n.(INTEGER(n) ⊃ (n gl s1=n gl s2)) ≡ s1=s2));;
```

AXIOM COMP:

```
   ∀f.             (FORM(f) ⊃ (●(f)=(LPARSYM c f) c RPARSYM),
   ∀f1 f2.         ((FORM(f1) ∧ FORM(f2)) ⊃ (f1 dis f2)=(●(f1) c ORSYM) c ●(f2)),
   ∀f1 f2.         ((FORM(f1) ∧ FORM(f2)) ⊃ (f1 impl f2)=(●(f1) c IMPSYM) c ●(f2)),
   ∀f.             (FORM(f) ⊃ neg(f)=(f impl FALSESYM)),
   ∀f1 f2.         ((FORM(f1) ∧ FORM(f2)) ⊃ (f1 con f2)=(●(f1) c ANDSYM) c ●(f2)),
   ∀x f2.          ((INDVAR(x) ∧ FORM(f2)) ⊃ (x gen f2)=(FORALLSYM c x ) c f2)    ,
   ∀x f2.          ((INDVAR(x) ∧ FORM(f2)) ⊃ (x ex f2)=(EXISTSYM c x) c f2)   ;;
```

2.4 Sequences

AXIOM SEQ:

```
   ∀sq.            (SEQUENCE(sq) ⊃ sq=scar(sq) cc scdr(sq)) ,
   ∀sq1 sq2.       ((SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃ (sq1=SLAMBDA ⊃
                      scar(sq1 cc sq2)=scar(sq2))),
   ∀sq1 sq2.       ((SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃ (sq1≠SLAMBDA ⊃
                      scar(sq1 cc sq2)=scar(sq1))),
   ∀sq1 sq2.       ((SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃ (sq1=SLAMBDA ⊃
                      scdr(sq1 cc sq2)=scdr(sq2))),
   ∀sq1 sq2.       ((SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃ (sq1≠SLAMBDA ⊃
                      scdr(sq1 cc sq2)=scdr(sq1) cc sq2)) ,
   ∀sq.            (SEQUENCE(sq) ⊃ sq cc SLAMBDA=SLAMBDA cc sq),
   ∀sq.            (SEQUENCE(sq) ⊃ sq cc SLAMBDA=sq),
   ∀sq1 sq2 sq3    ((SEQUENCE(sq1) ∧ SEQUENCE(sq2) ∧ SEQUENCE(sq3)) ⊃
                      (sq1 cc (sq2 cc sq3)=(sq1 cc sq2) cc sq3))
   ∀s.             (STRING(s) ⊃ (slen(s)=1 ∨ s=SLAMBDA)),
   ∀sq.            (SEQUENCE(sq) ⊃ slen(sq)≥0),
   ∀sq1 sq2.       ((SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃ slen(sq1 cc sq2)=slen(sq1)+slen(sq2)) ,
   ∀sq.            (SEQUENCE(sq) ⊃ 0 sgl sq=SLAMBDA),
   ∀sq.            (SEQUENCE(sq) ⊃ 1 sgl sq=scar(sq)),
   ∀n sq.          ((INTEGER(n) ∧ SEQUENCE(sq)) ⊃ ((n>1) ⊃ ((n sgl sq)=((n-1) sgl scdr(sq)))) ;;
```

AXIOM SUBSEQDEF:

```
   ∀n1 n2 sq1 sq2.  ((INTEGER(n1) ∧ INTEGER(n2) ∧ SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃
```

```
                    (SUBSEP(sq1,sq2,n1,n2) ≡ (slen(sq2)=n2-n1+1 ∧ (∀n.(n≥n1 ∧ n≤n2 ⊃
                    n sgl sq2=(n-n1+1) sgl sq1))))) ,
∀n1 n2 sq1 sq2.  ((INTEGER(n1) ∧ INTEGER(n2) ∧ SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃
                    (SUBSEP(sq1,sq2,n1,n2) ≡ subseq(sq1,n1,n2)=sq2)),
∀sq1 sq2.        ((SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃ (SUBSSE(sq1,sq2) ≡
                    ∃n1 n2.(SUBSEP(sq1,sq2,n1,n2))));;
```

AXIOM EQSQ:

```
∀sq1 sq2.        ((SEQUENCE(sq1) ∧ SEQUENCE(sq2)) ⊃ (∀n.(n sgl sq1=n sgl sq2) ⊃ sq1=sq2));;
```

2.5 Formulas

AXIOM FIND:

```
∀sq.        (FIND(0,LAMBDA,sq)≡ SEQUENCE(sq)),
∀n s sq.    (FIND(n,s,sq) ≡ INTEGER(n) ∧ STRING(s) ∧ SEQUENCE(sq) ∧
                ∃n s1 s2.(INTEGER(n) ∧ STRING(s1) ∧ STRING(s2) ∧ (0<s ∧ s<slen(sq)) ∧
                (s1=(n sgl sq)) ∧ (s=(s1 c s2)) ∧ FIND(n-1,s2,sq)));;
```

AXIOM FINDTOP:

```
∀sq.        (FINDTOP(0,SLAMBDA,sq)≡ SEQUENCE(sq)),
∀n s sq.    (FINDTOP(n,s,sq) ≡ INTEGER(n) ∧ STRING(s) ∧ SEQUENCE(sq) ∧
                ∃s1 s2.(STRING(s1) ∧ STRING(s2) ∧ (s1≠LAMBDA) ∧ (s=(s1 c s2)) ∧
                (s=scar(sq)) ∧ FINDTOP(n-1,s2,scar(sq))));;
```

AXIOM TERM:

```
∀sq.        (TERMSEQ(sq) ≡ SEQUENCE(sq) ∧ ((slen(sq)=1 ∧ INDVAR(1 sgl sq)) ∨
                (slen(sq)>1 ∧ TERMSEQ(scdr(sq)) ∧ (INDVAR(scar(sq)) ∨
                ∃n s.INTEGER(n) ∧ STRING(s) ∧ (s=car(scar(sq)) ∧ OPCONST(s) ∧ n=arity(s) ∧
                FIND(n,cdr(scar(sq)),scdr (sq))))))),
∀t.         (TERM(t) ≡ STRING(t) ∧ ∃sq.(TERMSEQ(sq) ∧ t=car(sq)));;
```

AXIOM WFF:

```
∀f.         (ELF(f) ≡ STRING(f) ∧ (f=FALSESYM ∨ PREDPAR0(f) ∨ ∃n sq.(INTEGER(n) ∧
                SEQUENCE(sq) ∧ PREDPAR(car(f)) ∧ n=arity(car(f)) ∧ TERMSEQ(sq) ∧
                        FINDTOP(n,cdr(f),sq)))),
∀sq.        (FRR(sq) ≡ SEQUENCE(sq) ∧ (sq≠SLAMBDA) ∧ (ELF(scar(sq)) ∨
                (FRR(scdr(sq)) ∧ ∃s1 s2.(STRING(s1) ∧ STRING(s2) ∧
                (((scar(sq)=neg(s1) ∧ FIND(1,x1,scdr(sq))) ∨
                (scar(sq)=(s1 dis s2) ∧ FIND(2,(s1 c s2),scdr(sq))) ∨
                (scar(sq)=(s1 con s2) ∧ FIND(2,(s1 c s2),scdr(sq))) ∨
                (scar(sq)=(s1 impl s2) ∧ FIND(2,(s1 c s2),scdr(sq))) ∨
                (scar(sq)=(s1 gen s2) ∧ INDVAR(s1) ∧ FIND(1,s2,scdr(sq))) ∨
                (scar(sq)=(s1 ex s2) ∧ INDVAR(s1) ∧ FIND(1,s2,scdr(sq))))))))),
∀f.         (FORM(f) ≡ STRING(f) ∧ ∃sq.(FRR(sq) ∧ f=scar(sq)));;
```

2.6 Free and bound variables and the substitution

AXIOM BOUNDV:

```
∀x n f.(GEB(x,n,f) ≡ INDVAR (x) ∧ INTEGER(n) ∧ FORM (f) ∧ ∃s1 s2 f1.(STRING(s1)∧
            FORM(f1) ∧ STRING(s2) ∧ len(s1)+1<n ∧ n<(len(f)-len(s2)) ∧
            (x=n gl f)∧((f=(s1 c ((x gen f1) c s2))) ∨ (f=(s1 c ((x ex f1) c f3))))));;
```

```
AXIOM FREEV:
    ∀x n f.     (FRN(x,n,f) ≡ INDVAR(x) ∧ INTEGER(n) ∧ FORM (f) ∧ x=(n gl f) ∧
                     ¬GEB(x,n,f)),
    ∀x f.       (FR(x,f) ≡ ∃n.(INTEGER(n) ∧ FRN(x,n,f)));;

AXIOM FIRSTFRDF:
    ∀x n f.     (FIRSTFREE(x,n,f) ≡ FRN(x,n,f) ∧ ∀n1.(INTEGER(n1) ∧ x=n1 gl f ⊃
                     (n1≥n ∨ GEB(x,n1,f))))),
    ∀x n f.     (FIRSTFREE(x,n,a) ≡ firstfree(x,f)=n);;

AXIOM  KFREEOCCDF:
    ∀x k n f.   (KTHFREEOCC(x,k,n,f) ≡ (INDVAR(x) ∧ INTEGER(k) ∧ INTEGER(n) ∧
                     FORM(f) ∧ (k=0 ∧ n=0) ∨
                     (n=len(f) ∧ ∀n2.((INTEGER(n2) ∧ n2>kthfreeocc(x,k-1,f)) ⊃ ¬FRN(x,n2,f))) ∨
                     (FRN(x,n,f) ∧ ∀n1.((INTEGER(n1)∧(n1<k ∧ n1>0)) ⊃
                              ∃n2.(INTEGER(n2) ∧ n2<n ∧ KTHFREEOCC(x,n1,n2,f)))))),
    ∀x k n f.   (KTHFREEOCC(x,k,n,f) ≡ kthfreeocc(x,k,f)=n),
    ∀x k n f.   (KTHFREEOCC(x,k,n,f) ⊃ numbfreeocc(x,n,f)=k),
    ∀x k n f.   (numbfreeocc(x,n,f)=k ⊃ (KTHFREEOCC(x,k,n,f) ∨
                     (n<kthfreeocc(x,k,f) ∧ n>kthfreeocc(x,k-1,f)));;

AXIOM SUBSTDF:
    ∀x t f1 f2.  (SBT(x,t,f1,f2) ≡ ((INDVAR(x) ∧ TERM(t) ∧ FORM(f1) ∧ FORM(f2)) ⊃
                      ∀n1 n2.((INTEGER(n1) ∧ INTEGER(n2) ∧
                      n2=numbfreeocc(x,n1,f1)*(len(t)-1))+n1 ⊃
                      ((¬INDVAR(n1 gl f1) ⊃ n1 gl f1 = n2 gl f2) ∧
                      (INDVAR(n1 gl f1) ⊃ ((FRN(x,n1,f1) ⊃ SUBT(t,f2,n2)) ∧
                               (¬FRN(x,n1,f1) ⊃ INVART(n1,f1,n2,f2)))))))),
    ∀t f2 n2.    (SUBT(t,f2,n2) ≡ (TERM(t) ∧ FORM(f2) ∧ INTEGER(n2) ∧
                      ∀x2 k.((INDVAR(x2) ∧ INTEGER(k) ∧ ((k gl t)=x2)) ⊃
                      FRN(x2,n2-(len(t)-k),f2)))),
    ∀n1 f1 n2 f2.(INVART(n1,f1,n2,f2) ≡ (INTEGER(n1) ∧ FORM(f1) ∧ INTEGER(n2) ∧
                      FORM(f2) ∧ (GEB(n2 gl f2,n2,f2) ≡ GEB(n1 gl f1,n1,f1)) ∧
                      (FRN(n2 gl f2,n2,f2) ≡ FRN(n1 gl f1,n1,f1)) ∧
                      n2 gl f2=n1 gl f1)),
    ∀x t f1 f2.  ((INDVAR(x) ∧ TERM(t) ∧ FORM(f1) ∧ FORM(f2)) ⊃
                      (SBT(x,t,f1,f2)≡sbt(x,t,f1)=f2)),
    ∀x t f1.     ((INDVAR(x) ∧ TERM(t) ∧ FORM(f1)) ⊃ FORM(sbt(x,t,f1)));;

AXIOM SUBDEF:
    ∀x1 x2 f1 f2.(SBV(x1,x2,f1,f2) ≡ ((INDVAR(x1) ∧ INDVAR(x2) ∧ FORM(f1) ∧ FORM(f2)) ⊃
                      ∀n.(INTEGER(n) ⊃ ((¬INDVAR(n gl f1) ⊃ n gl f1=n gl f2) ∧
                      (INDVAR(n gl f1) ⊃ ((FRN(x1,n,f1) ⊃ FRN(x2,n,f2)) ∧
                      (¬FRN(x1,n,f1) ⊃ INVARV(n,f1,f2)))))))),
    ∀n f1 f2.    (INVARV(n,f1,f2) ≡ (INTEGER(n) ∧ FORM(f1) ∧ FORM(f2) ∧
                      (GEB(n gl f2,n,f2) ≡ GEB(n gl f1,n,f1)) ∧
                      FRN(n gl f2,n,f2) ≡ FRN(n gl f1,n,f1)) ∧
                      n gl f2=n gl f1)),
    ∀x1 x2 f1 f2 .((INDVAR(x1) ∧ INDVAR(x2) ∧ FORM(f1) ∧ FORM(f2)) ⊃
                      (SBV(x1,x2,f1,f2) ≡ sbv(x1,x2,f1)=f2)),
    ∀x1 x2 f1.   ((INDVAR(x1) ∧ INDVAR(x2) ∧ FORM(f1)) ⊃ FORM(sbv(x1,x2,f1)));;
```

2.7 Rules of inference

```
AXIOM ANDIRUL:
      ∀sq pf1 pf2.       (ANDI(sq,pf1,pf2) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf1) ∧ PROOFTREE(pf2) ∧
                              ∃f1 f2.(scdr(sq)=(pf1 cc pf2) ∧ scar(sq)=f1 con f2 ∧ FORM(f1) ∧
                              FORM(f2) ∧ f1=scar(pf1) ∧ f2=scar(pf2)))),
      ∀sq pf.            (ANDE(sq,pf) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf) ∧ ∃f1.(scdr(sq)=pf ∧
                              FORM(f1) ∧ (((scar(sq) con f1)=scar(pf)) ∨
                              ((f1 con (scar(sq))=scar(pf))))));;

AXIOM FALSERUL :
      ∀sq pf1 pf2.       (FALSEI(sq,pf1,pf2) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf1) ∧ PROOFTREE(pf2) ∧
                              ∃f1.((scdr(sq)=(pf1 cc pf2)) ∧ (scar(sq)=FALSESYM) ∧ FORM(f1) ∧
                              (neg(x)=scar(pf1)) ∧ (x1=scar(pf2))))),
      ∀sq pf.            (FALSEE(sq,pf) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf) ∧ (scar(pf)=FALSESYM) ∧
                              scdr(sq)=pf));;

AXIOM IMPLRUL :
      ∀sq pf1 pf2.       (IMPLE(sq,pf1,pf2) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf1) ∧
                              PROOFTREE(pf2) ∧ ∀f1.((scdr(sq)=(pf1 cc pf2)) ∧ FORM(f1) ∧
                              (scar(pf1)=(f1 impl (scar(sq))) ∧ (scar(pf2)=f1)))),
      ∀sq pf f1.         (IMPLID(sq,pf,f1) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf) ∧ scdr(sq)=pf ∧
                              FORM(f1) ∧ ∃f2.((scar(sq)=(f1 impl x2)) ∧ FORM(f1) ∧ (f2=scar(pf)) ∧
                              ∃n.(INTEGER(n) ∧ f1=(n sgl pf))))));;
      ∀sq pf.            (IMPLI(sq,pf) ≡ ∃f.IMPLID(sq,pf,f));;

AXIOM NEGRUL:
      ∀sq pf f1.         (NOTID(sq,pf,f1) ≡ (scdr(sq)=pf ∧ SEQUENCE(sq) ∧ PROOFTREE(pf) ∧
                              FORM(f1) ∧ ∃n.((scar(pf)=FALSESYM) ∧ scar(sq)=neg(f1) ∧
                              INTEGER(n) ∧ ((n sgl pf)=f1)))),
      ∀sq pf.            (NOTI(sq,pf) ≡ ∃f.NOTID(sq,pf,f)),
      ∀sq pf f1.         (NOTED(sq,pf,f1) ≡ (scdr(sq)=pf ∧ SEQUENCE(sq) ∧ PROOFTREE(pf) ∧
                              FORM(f) ∧ ∃n.((scar(pf)=FALSESYM) ∧ INTEGER(n) ∧
                              ((n sgl pf)=neg(scar(sq)))))),
      ∀sq pf.            (NOTE(sq,pf) ≡ ∃f.NOTED(sq,pf,f));;

AXIOM ORRUL:
      ∀sq pf.            (ORI(sq,pf) ≡ (scdr(sq)=pf ∧ SEQUENCE(sq) ∧ PROOFTREE(pf) ∧
                         ∃f1 f2.((scar(sq) ≡ (f1 dis f2)) ∧ FORM(f1) ∧ FORM(f2) ∧ (f1=scar(pf)) ∨
                         (f2=scar(pf))))),
      ∀sq pf1 pf2 pf3 f1 f2.(ORED(sq,pf1,pf2,pf3,f1,f2) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf1) ∧
                         PROOFTREE(pf2) ∧ PROOFTREE(pf3) ∧ FORM(f1) ∧ FORM(f2) ∧
                         (scdr(sq)=(pf1 cc (pf2 cc pf3)) ∧
                         (scar(pf1)=(f1 dis f2))∧(scar(pf2)=scar(sq)) ∧ (scar(pf3)=scar(sq)) ∧
                         ∃n1.(n1 sgl pf2)=f1) ∧ ∃n1.(n1 sgl pf3)=f2)))),
      ∀sq pf1 pf2 pf3.  (ORE(sq,pf1,pf2,pf3) ≡ ∃f1 f2.ORED(sq,pf1,pf2,pf3,f1,f2));;

AXIOM EXRUL :
      ∀sq pf x t.        (EXI(sq,pf,x,t) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf) ∧ INDVAR(x) ∧ TERM(t) ∧
                         ∃f1.((scdr(sq)=pf1) ∧ (scar(sq)=(x ex f1)) ∧ FORM(f1) ∧
                         scar(pf)=sbt(x,t,f1)))),
      ∀sq pf1 pf2 x1 x2 f1.(EXED(sq,pf1,pf2,x1,x2,f1) ≡ (SEQUENCE(sq) ∧ PROOFTREE(pf1) ∧
                         INDVAR(x1) ∧ INDVAR(x2) ∧ (scdr(sq)=(pf1 cc pf2)) ∧ FORM(f1) ∧
                         (scar(pf1)=(x1 ex f1)) ∧ (scar(sq)= scar(pf2)) ∧
                         ∃n.((n sgl pf2)=sbt(x1,x2,f1) ∧ INTEGER(n) ∧ EXAPPL(x2,pf2,f1))))),
      ∀sq pf1 pf2 x1 x2.(EXE(sq,pf1,pf2,x1,x2) ≡ EXED(sq,pf1,x1,x2)),
      ∀x pf f.           (EXAPPL(x,pf,f) ≡ (INDVAR(x) ∧ PROOFTREE(pf) ∧ FORM(f) ∧ ¬FR(x,scar(pf)) ∧
```

```
                        ¬FR(x,f) ∧ ∀f1.(DEPEND(pf,f1) ⊃ ¬FR(x,f1))));;
AXIOM GENRUL:
      ∀sq sq1 x t.      (GENE(sq,sq1,x,t) ≡ (SEQUENCE(sq) ∧ INDVAR(x) ∧ TERM(t) ∧ scdr(sq)=sq1 ∧
                          PROOFTREE(sq1) ∧ ∃f.(FORM(f) ∧ scar(sq1)=x gen f ∧
                          scar(sq)=sbt(x,t,f)))),
      ∀sq sq1 x1 x2.    (GENI(sq,sq1,x1,x2) ≡ (SEQUENCE(sq) ∧ INDVAR(x1) ∧ INDVAR(x2) ∧
                          scdr(sq)=sq1 ∧ PROOFTREE(sq1) ∧ ∃f.(FORM(f) ∧ (scar(sq)=x1 gen f) ∧
                          scar(sq1)=sbt(x1,x2,f) ∧ APGENI(x2,sq1)))),
      ∀x sq.            (APGENI(x,sq) ≡ (INDVAR(x) ∧ ∀f.(DEPEND(sq,f) ⊃ ¬FR(x,f))) ∧
                          PROOFTREE(sq)),
      ∀sq.              (PROOFTREE(sq) ⊃ ∃x.(INDVAR(x) ∧ APGENI(x,sq))) ;;
```

2.8 Deduction

```
AXIOM PROOF:
      ∀sq.              (PROOFTREE(sq) ≡ ((SEQUENCE(sq) ∧ FORM(sq)) ∨
                          ∃pf.(PROOFTREE(pf) ∧ (ORI(sq,pf) ∨ ANDE(sq,pf) ∨ FALSEE(sq,pf) ∨
                            NOTI(sq,pf) ∨ NOTE(sq,pf) ∨ IMPLI(sq,pf))) ∨
                          ∃pf x t .(PROOFTREE(pf) ∧ INDVAR(x) ∧ TERM(t) ∧
                            (GENI(sq,pf,x,t) ∨ GENE(sq,pf,x,t) ∨ EXI(sq,pf,x,t))) ∨
                          ∃pf1 pf2.(PROOFTREE(pf1) ∧ PROOFTREE(pf2) ∧
                            (ANDI(sq,pf1,pf2) ∨ FALSEI(sq,pf1,pf2) ∨ IMPLE(sq,pf1,pf2))) ∨
                          ∃pf1 pf2 x1 x2.(PROOFTREE(pf1) ∧ PROOFTREE(pf2) ∧ INDVAR(x1) ∧
                            INDVAR(x2) ∧ EXE(sq,pf1,pf2,x1,x2)) ∨
                          ∃pf1 pf2 pf3.(PROOFTREE(pf1) ∧ PROOFTREE(pf2) ∧ PROOFTREE(pf3) ∧
                            ORE(sq,pf1,pf2,pf3)))) ;;
AXIOM DEPNDG:
      ∀sq f.            (DEPEND(sq,f) ⊃ (SEQUENCE(sq) ∧ FORM(f) ∧ SUBSSE(f,sq))),
      ∀sq f.            ((SEQUENCE(sq) ∧ FORM(f) ∧ sq=f) ⊃ DEPEND(sq,f)) ;;
AXIOM DEPEND:
      ∀pf pf1 f.        (((PROOFTREE(pf) ∧ PROOFTREE(pf1) ∧ (pf1=scdr(pf))) ⊃
                          (DEPEND(pf,f) ≡ DEPEND(pf1,f))) ≡
                            (ORI(pf,pf1) ∨ ANDE(pf,pf1) ∨ FALSEE(pf,pf1) ∨
                            ∃f1.(FORM(f1) ∧ (NOTID(pf,pf1,f1) ∨ NOTED(pf,pf1,f1) ∨
                              IMPLID(pf,pf1,f1)) ∧ f1≠f) ∨
                            ∃x t .(INDVAR(x) ∧ TERM(t) ∧ GENI(pf,pf1,x,t) ∨
                              GENE(pf,pf1,x,t) ∨ EXI(pf,pf1,x,t))))) ;;
AXIOM DEP:
      ∀pf pf1 pf2 f.    (((PROOFTREE(pf) ∧ PROOFTREE(pf1) ∧ PROOFTREE(pf2) ∧
                          ((pf1 cc pf2=scdr(pf)) ∨ (pf2 cc pf1=scdr(pf)))) ⊃ (DEPEND(pf,f) ≡
                          (DEPEND(pf1,f) ∨ DEPEND(pf2,f)))) ≡ (ANDI(pf,pf1,pf2) ∨
                          FALSEI(pf,pf1,pf2) ∨ IMPLE(pf,pf1,pf2) ∨
                          ∃x1 x2 f1.(EXED(pf,pf1,pf2,x1,x2,f1) ∧ f≠f1)));;
AXIOM DEPND:
      ∀pf pf1 pf2 pf3 f. (((PROOFTREE(pf) ∧ PROOFTREE(pf1) ∧
                         PROOFTREE(pf2) ∧ PROOFTREE(pf3) ∧
                           (((pf1 cc (pf2 cc pf3))=scdr(pf)) ∨
                            ((pf1 cc (pf3 cc pf2))=scdr(pf)) ∨
```

```
                         ((pf2 cc (pf1 cc pf3))=scdr(pf)) ∨
                         ((pf2 cc (pf3 cc pf1))=scdr(pf)) ∨
                         ((pf3 cc (pf1 cc pf2))=scdr(pf)) ∨
                         ((pf3 cc (pf2 cc pf1))=scdr(pf)))) ⊃
                         (DEPEND(pf,f) ≡ (DEPEND(pf1,f) ∨ DEPEND(pf2,f) ∨ DEPEND(pf3,f)))) ≡
                           ∃f1 f2.(ORED(pf,pf1,pf2,pf3,f1,f2) ∧ f≠f1 ∧ f≠f2)) ,

AXIOM NDEPND:
      ∀pf1 pf2 f.        ((NOTID(pf1,pf2,f) ∨ NOTED(pf1,pf2,f) ∨ IMPLID(pf1,pf2,f)) ⊃
                            ¬DEPEND(pf1,f)),
      ∀pf1 pf2 pf3 x1 x2 f(EXED(pf1,pf2,pf3,x1,x2,f) ⊃ ¬ DEPEND(pf1,f)) ,
      ∀pf1 pf2 pf3 pf4 f1 f2. (ORED(pf1,pf2,pf3,pf4,f1,f2) ⊃ ¬ DEPEND(pf1,f1) ∧ ¬DEPEND(pf1,f2));;

AXIOM PROVABLE:
      ∀f.                (BEW(f) ≡ FORM(f) ∧ ∃sq.(PROOFTREE(sq) ∧ f=scar(sq) ∧
                            ∀f1.(DEPEND(sq,f1) ⊃ AXIOM(f1))));;

AXIOM THEORY:
      ∀x f.              (AXIOM(f) ⊃ ¬ FR(x,f)∧FORM(f));;

AXIOM INFVAR:
      ∀s.∃x.∀n.          n gl s≠x) ;;
```

APPENDIX 3

THE PROOF OF "IF f IS A WFF ALSO . x.f IS A WFF"

3.1 FOL commands and printout in the many sorted logic

commands

```
∀E WFF1, x gen f;
TAUTEQ (x gen f= x gen f) ∨ (x gen f = x ex f);
UNIFY --;#2#2#1 , -;
TAUT ---;#1, 1:-;
```

proof

```
1  FORM(x gen f)=(ELF(x gen f)∨(∃x1 f1.((x gen f)=(x1 gen f1)∨(x gen f)=(x1 ex f1))∨
   (∃f1 f2.((x gen f)=(f1 dis f2)∨((x gen f)=(f1 con f2)∨(x gen f)=(f1 impl f2)))∨
   ∃f1.(x gen f)=neg(f1))))

2  (x gen f)=(x gen f)∨(x gen f)=(x ex f)

3  ∃x1 f1.((x gen f)=(x1 gen f1)∨(x gen f)=(x1 ex f1))

4  FORM(x gen f)
```

3.2 FOL commands in the earlier axiomatization

```
DECLARE INDVAR  A U;
label  hpt1;
ASSUME  FORM(f) ∧ INDVAR (x1) ;
label  teo1 ;
ASSUME  ∀f s.(SEQUENCE(sq)∧ sq ≠ SLAMBDA ⊃ (STRING(s)⊃ (s cc sq) ≠ SLAMBDA));
label  teo2 ;
ASSUME  ∀s sq.(STRING(s)∧SEQUENCE(sq)⊃ scar(s cc sq)= s);
label  teo3 ;
ASSUME  ∀s sq.(STRING(s)∧SEQUENCE(sq)⊃ scdr(s cc sq)= sq);
label  teo4 ;
ASSUME  ∀sq.(SEQUENCE(sq)∧sq≠SLAMBDA ⊃ find(1,scar(sq),sq));
label  teo5 ;
ASSUME  ∀f x.(FORM(f)∧INDVAR(x) ⊃STRING(x gen f));
label  teo6;
ASSUME  ∀s sq.(STRING(s)∧SEQUENCE(sq) ⊃SEQUENCE(s cc sq));
label  teo7 ;
ASSUME  ∀x.(INDVAR(x) ⊃ STRING(x));

∀e WFF2 f ;
LABEL ass1 ;
taut ∃sq.(FRR(sq)∧f=scar(sq)) 1:-;
ASSUME FRR(SQ) ∧ f = SCAR(SQ) ;
∀e WFF1  SQ ;
∀e teo1 SQ ,x1 gen f;
```

```
∀e teo2 x1 gen f ,SQ;
∀e teo3 x1 gen f ,SQ;
∀e teo4          SQ;
∀e teo5 f ,x1;
∀e teo7 x1;
∀e Wff1 (x1 gen f) cc SQ;

TAUTEQ -:*2*2*2*2*2*1*1[s1←f : s2←x1] 1:-;
unify --:*2*2*2*2*2 -;
∀e teo6 x1 gen f ,SQ;
∀e WFF2 x1 gen f ;
tauteq -:*2*2*1[sq←(x1 gen f) cc SQ] 1:-;
unify --:*2*2 -;
taut FORM(x1 gen f) 1:-;
∃e ass1,-,SQ;
⊃i hpt1,-;
∀I -,x1,f;
```

3.3 Printout of the proof in the earlier axiomatization

1 FORM(f)∧INDVAR(x1) (1) --- ASSUME

2 ∀sq s.((SEQUENCE(sq)∧sq≠SLAMBDA)⊃(STRING(s)⊃(s cc sq)≠SLAMBDA)) (2) --- ASSUME

3 ∀s sq.((STRING(s)∧SEQUENCE(sq))⊃scar(s cc sq)=s) (3) --- ASSUME

4 ∀s sq.((STRING(s)∧SEQUENCE(sq))⊃scdr(s cc sq)=sq) (4) --- ASSUME

5 ∀sq.((SEQUENCE(sq)∧sq≠SLAMBDA)⊃find(1,scar(sq),sq)) (5) --- ASSUME

6 ∀f x.((FORM(f)∧INDVAR(x))⊃STRING(x gen f)) (6) --- ASSUME

7 ∀s sq.((STRING(s)∧SEQUENCE(sq))⊃SEQUENCE(s cc sq)) (7) --- ASSUME

8 ∀x.(INDVAR(x)⊃STRING(x)) (8) --- ASSUME

9 FORM(f)≡(STRING(f)∧∃sq.(FRR(sq)∧f=scar(sq))) --- ∀E WFF2 f

10 ∃sq.(FRR(sq)∧f=scar(sq)) (1 2 3 4 5 6 7 8) --- TAUT 1:9

11 FRR(SQ)∧f=scar(SQ) (11) --- ASSUME

12 FRR(SQ)≡(SEQUENCE(SQ)∧(SQ≠SLAMBDA∧(ELF(scar(SQ))∨(FRR(scdr(SQ))∧∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar(SQ)=NEG(s1)∧find(1,s1,scdr(SQ)))∨((scar(SQ)=(s1 dis s2)∧find(2,s1 c x2,scdr(SQ)))∨((scar(SQ)=(s1 con s2)∧find(2,s1 c s2,scdr(SQ)))∨((scar(SQ)=(s1 impl s2)∧find(2,s1 c s2,scdr(SQ)))∨((scar(SQ)=(s1 gen s2)∧(INDVAR(s1)∧find(1,s2,scdr(SQ))))∨(scar(SQ)=(s1 ex s2)∧(INDVAR(s1)∧find(1,s2,scdr(SQ))))))))))))))) --- ∀E WFF1 SQ

13 (SEQUENCE(SQ)∧SQ≠SLAMBDA)⊃(STRING(x1 gen f)⊃((x1 gen f) cc SQ)≠SLAMBDA) (2) --- ∀E 2 SQ,x1 gen f

```
14 (STRING(x1 gen f)∧SEQUENCE(SQ))⊃scar((x1 gen f) cc SQ)=(x1 gen f)
   (3) --- ∀E 3 x1 gen f,SQ

15 (STRING(x1 gen f)∧SEQUENCE(SQ))⊃scdr((x1 gen f) cc SQ)=SQ (4) --- ∀E 4 x1 gen f,SQ

16 (SEQUENCE(SQ)∧SQ≠SLAMBDA)⊃find(1,scar(SQ),SQ) (5) --- ∀E 5 SQ

17 (FORM(f)∧INDVAR(x1))⊃string(x1 gen f) (6) --- ∀E 6 f,x1

18 INDVAR(x1)⊃STRING(x1) (8) --- ∀E 8 x1

19 FRR((x1 gen f) cc SQ)≡(SEQUENCE((x1 gen f) cc SQ)∧(((x1 gen f) cc U)≠
   SLAMBDA∧(ELF(scar((x1 gen f) cc SQ))∨(FRR(scdr((x1 gen f) cc SQ))∧
   ∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar((s1 gen f) cc SQ)=NEG(s1)∧
   find(1,s1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 dis s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 con s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 impl s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 gen s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(s1 ex s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ))))))))))))))) --- ∀E WFF1 (x1 gen f) cc SQ

20 STRING(x1)∧(STRING(f)∧((scar((x1 gen f) cc SQ)=NEG(x1)∧
   find(1,x1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 dis f)∧
   find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 con f)∧
   find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 impl f)∧
   find(2,x1 c f,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(x1 gen f)∧(INDVAR(x1)∧
   find(1,f,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(x1 ex f)∧(INDVAR(x1)∧
   find(1,f,scdr((x1 gen f) cc SQ)))))))))) (1 2 3 4 5 6 7 8 11) --- TAUTEQ 1:19

21 ∃s1 s2.(STRING(s1)∧(STRING(s2)∧((scar((s1 gen f) cc SQ)=NEG(s1)∧
   find(1,s1,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 dis s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 con s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 impl s2)∧
   find(2,s1 c s2,scdr((x1 gen f) cc SQ)))∨((scar((x1 gen f) cc SQ)=(s1 gen s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ))))∨(scar((x1 gen f) cc SQ)=(s1 ex s2)∧(INDVAR(s1)∧
   find(1,s2,scdr((x1 gen f) cc SQ))))))))))))))) (1 2 3 4 5 6 7 8 11) --- UNIFY 20

22 (STRING(x1 gen f)∧SEQUENCE(SQ))⊃SEQUENCE((x1 gen f) cc SQ) (7) ---∀E 7 x1 GEN f,SQ

23 FORM(x1 gen f)≡(STRING(x1 gen f)∧∃sq.(FRR(sq)∧(x1 gen f)=scar(sq))) --- ∀E WFF2 x1 gen f

24 FRR((x1 gen f) cc SQ)∧(x1 gen f)=scar((x1 gen f) cc SQ) (1 2 3 4 5 6 7 8 11) TAUTEQ 1:23

25 ∃sq.(FRR(sq)∧(x1 gen f)=scar(sq)) (1 2 3 4 5 6 7 8 11) --- UNIFY 24

26 FORM(x1 gen f) (1 2 3 4 5 6 7 8 11) --- TAUT 1:25

27 FORM(x1 gen f) (1 2 3 4 5 6 7 8) --- ∃E 10 26 U

28 (FORM(f)∧INDVAR(x1))⊃FORM(x1 gen f) (2 3 4 5 6 7 8) --- ⊃I 1 27

29 ∀f x1.((FORM(f)∧INDVAR(x1))⊃FORM(x1 gen f)) (2 3 4 5 6 7 8) --- ∀I 28 x1←x1 f←f
```

APPENDIX 4

THE PROOF OF THE EQUIVALENCE BETWEEN SBV AND SBT FOR VARIABLES

4.1 FOL commands in the many sorted logic

```
LABEL ARITH1;  ASSUME ∀n x.(n*(len(x)-1)=0);
LABEL ARITH2;  ASSUME ∀n.  (0+n=n);
LABEL ARITH3;  ASSUME ∀x.  (len(x)-1)=0;
LABEL ARITH4;  ASSUME ∀n.  (n-0)=n;
LABEL STRING1; ASSUME ∀x.  1 gl x = x;
```

Proof of the First Lemma: ∀x f n.(SUBT(x,f,n) ⊃ FRN(x,n,f))

```
LABEL HPTLEM; ASSUME SUBT(x,f,n);
∀e SUBSTDF1,x,f,n;
TAUT -:#2,--,-;
∀e -,x,1;
∀e STRING1,x; substr - in --;
∀e ARITH3 ,x; substr - in --;
∀e ARITH4 ,n; substr - in --;
TAUTEQ FRN(x,n,f),HPTLEM+1:-;
⊃I HPTLEM,-;
LABEL LEMMA1;∀I -,x,f,n;
```

Proof of the Second Lemma: ∀n f1 f2.(INVART(n,f1,n,f2) ≡ INVARV(n,f1,f2))

```
∀e SUBSTDF2,n,f1,n,f2;
∀e SUBDEF1 ,n,f1,f2;
TAUT --:#1 ≡ -:#1,--,-;
LABEL LEMMA2; ∀i -,n,f1,f2;
```

Proof of the Main Theorem: ∀x1 x2 f1 f2.(SBT(x1,x2,f1,f2)⊃SBV(x1,x2,f1,f2))

```
LABEL HPT; ASSUME SBT(x1,x2,f1,f2);
∀e SUBSTDF0,x1,x2,f1,f2;
TAUT -:#2,HPT,-;
∀E -,n1,n1;
∀e ARITH1,numbfreeocc(x1,n1,f1),x2; substr - in --;
∀e ARITH2,n1;
∀e SUBDEF0 x1,x2,f1,f2;
∀e LEMMA1,x2,f2,n1;
∀e LEMMA2,n1,f1,f2;
TAUTEQ ---:#2#1[n←n1],HPT+1:-;
∀i -,n1←n;
TAUTEQ -----:#1,HPT+1:-;
⊃I HPT,-;
∀I -,x1,x2,f1,f2;
```

4.2 Printout of the proof in the many sorted logic

```
1  ∀n x.(n*(len(x)-1))=0  (1)

2  ∀n.(0+n)=n  (2)

3  ∀x.(len(x)-1)=0  (3)

4  ∀n.(n-0)=n  (4)

5  ∀x.(1 gl x)=x  (5)

6  SUBT(x,f,n)  (6)

7  SUBT(x,f,n)≡∀x2 k.((k gl x)=x2⊃FRN(x2,n-(len(x)-k),f))

8  ∀x2 k.((k gl x)=x2⊃FRN(x2,n-(len(x)-k),f))  (6)

9  (1 gl x)=x⊃FRN(x,n-(len(x)-1),f)  (6)

10  (1 gl x)=x  (5)

11  x=x⊃FRN(x,n-(len(x)-1),f)  (5 6)

12  (len(x)-1)=0  (3)

13  x=x⊃FRN(x,n-0,f)  (3 5 6)

14  (n-0)=n  (4)

15  x=x⊃FRN(x,n,f)  (3 4 5 6)

16  FRN(x,n,f)  (3 4 5 6)

17  SUBT(x,f,n)⊃FRN(x,n,f)  (3 4 5)

18  ∀x f n.(SUBT(x,f,n)⊃FRN(x,n,f))  (3 4 5)

19  INVART(n,f1,n,f2)≡((GEB(n gl f2,n,f2)≡GEB(n gl f1,n,f1))∧((FRN(n gl f2,n,f2)≡
    FRN(n gl f1,n,f1))∧(n gl f2)=(n gl f1)))

20  INVARV(n,f1,f2)≡((GEB(n gl f2,n,f2)≡GEB(n gl f1,n,f1))∧((FRN(n gl f2,n,f2)≡
    FRN(n gl f1,n,f1))∧(n gl f2)=(n gl f1)))

21  INVART(n,f1,n,f2)≡INVARV(n,f1,f2)

22  ∀n f1 f2.(INVART(n,f1,n,f2)≡INVARV(n,f1,f2))

23  SBT(x1,x2,f1,f2)  (23)

24  SBT(x1,x2,f1,f2)≡∀n1 n2.(n2=((numbfreeocc(x1,n1,f1)*(len(x2)-1))+n1)⊃
    ((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n2 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃
    SUBT(x2,f2,n2))∧(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n2,f2)))))
```

25 ∀n1 n2.(n2=((numbfreeocc(x1,n1,f1)*(len(x2)-1))+n1)⊃((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n2 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃SUBT(x2,f2,n2))∧(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n2,f2)))))) (23)

26 n1=((numbfreeocc(x1,n1,f1)*(len(x2)-1))+n1)⊃((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃SUBT(x2,f2,n1))∧(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n1,f2))))) (23)

27 (numbfreeocc(x1,n1,f1)*(len(x2)-1))=0 (1)

28 n1=(0+n1)⊃((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃SUBT(x2,f2,n1))∧(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n1,f2))))) (1 23)

29 (0+n1)=n1 (2)

30 SBV(x1,x2,f1,f2)≡∀n ((¬INDVAR(n gl f1)⊃(n gl f1)=(n gl f2))∧(INDVAR(n gl f1)⊃((FRN(x1,n,f1)⊃FRN(x2,n,f2))∧(¬FRN(x1,n,f1)⊃INVARV(n,f1,f2)))))

31 SUBT(x2,f2,n1)⊃FRN(x2,n1,f2) (3 4 5)

32 INVART(n1,f1,n1,f2)≡INVARV(n1,f1,f2)

33 (¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃FRN(x2,n1,f2))∧(¬FRN(x1,n1,f1)⊃INVARV(n1,f1,f2)))) (1 2 3 4 5 23)

34 ∀n.((¬INDVAR(n gl f1)⊃(n gl f1)=(n gl f2))∧(INDVAR(n gl f1)⊃((FRN(x1,n,f1)⊃FRN(x2,n,f2))∧(¬FRN(x1,n,f1)⊃INVARV(n,f1,f2))))) (1 2 3 4 5 23)

35 SBV(x1,x2,f1,f2) (1 2 3 4 5 23)

36 SBT(x1,x2,f1,f2)⊃SBV(x1,x2,f1,f2) (1 2 3 4 5)

37 ∀x1 x2 f1 f2.(SBT(x1,x2,f1,f2)⊃SBV(x1,x2,f1,f2)) (1 2 3 4 5)

4.3 FOL commands in the earlier axiomatization

```
LABEL ARITH1;  ASSUME ∀n x.((INTEGER(n) ∧ INDVAR(x))⊃(n*(len(x)-1)=0));
LABEL ARITH2;  ASSUME ∀n.  (INTEGER(n) ⊃ (0+n=n));
LABEL ARITH3;  ASSUME ∀x.  (INDVAR(x) ⊃ ((len(x)-1)=0));
LABEL ARITH4;  ASSUME ∀n.  (INTEGER(n) ⊃ (n-0)=n);
LABEL STRING1; ASSUME ∀x.  (INDVAR(x) ⊃ 1 gl x=x);
```

Proof of the First Lemma:
∀x n f.((INDVAR(x) ∧ INTEGER(n) ∧ FORM(f) ∧ SUBT(x,f,n)) ⊃ FRN(x,n,f))

```
LABEL HPTLEM; ASSUME INDVAR(x)∧FORM(f)∧INTEGER(n)∧SUBT(x,f,n);
LABEL FACT ; ASSUME INTEGER(1);
∀e SUBSTDF1,x,f,n;
TAUT -:*2*2*2*2,---,-;
∀e -,x,1;
∀e STRING1,x; TAUT -:*2,HPTLEM:-;substr - in ---;
∀e ARITH3,x; TAUT -:*2,HPTLEM:-;substr - in ---;
```

```
∀e ARITH4,n; TAUT -:#2,HPTLEM:-;substr - in ---;
TAUTEQ FRN(x,n,f),HPTLEM:-;
⊃I HPTLEM,-;
LABEL LEMMA1;∀I -,x,f,n;
```

Proof of the Second lemma : ∀k f1 f2.(INVART(k,f1,k,f2) ≡ INVARV(k,f1,f2))

```
∀e SUBSTDF2,k,f1,k,f2;
∀e SUBDEF1 ,k,f1,f2;
TAUT --:#1 ≡ -:#1,--,-;
LABEL LEMMA2; ∀i -,k,f1,f2;
```

Proof of the Main Theorem

```
∀x1 x2 f1 f2.((INDVAR(x1) ∧ INDVAR(x2) ∧ FORM(f1) ∧ FORM(f2) ∧ SBT(x1,x2,f1,f2)) ⊃
            SBV(x1,x2,f1,f2))

LABEL HPT; ASSUME INDVAR(x1)∧INDVAR(x2)∧FORM(f1)∧FORM(f2)∧SBT(x1,x2,f1,f2);

LABEL THTERM; ASSUME ∀x2.(INDVAR(x2)⊃ TERM(x2));
∀E THTERM,x2;
LABEL THNFRO; ASSUME ∀x1 n1 f1.INTEGER(numbfreeocc(x1,n1,f1));

∀e SUBSTDF0,x1,x2,f1,f2;
TAUT -:#2#2#2#2#2,HPT:-;
∀E -,n1,n1;

LABEL AUX;ASSUME INTEGER(n1);
∀E THNFRO,x1,n1,f1;

∀e ARITH1,numbfreeocc(x1,n1,f1),x2; TAUT -:#2,HPT:-;substr - in -----;
∀e ARITH2,n1;TAUT -:#2,HPT:-;SUBSTR-IN ---;
TAUTEQ -:#2,HPT:-;
∀e SUBDEF0 x1,x2,f1,f2;
∀e LEMMA1,x2,f2,n1;
∀e LEMMA2,n1,f1,f2;

TAUTEQ ---:#2#2#1#2[n←n1], HPT :-;
⊃I AUX,-;
∀i -,n1;
TAUTEQ -------:#1, HPT:-;
⊃I HPT,-;
∀I -,x1,x2,f1,f2;
```

4.4 Printout of the proof in the earlier axiomatization

```
1  ∀n x.((INTEGER(n)∧INDVAR(x))⊃(n*(len(x)-1))=0)  (1)

2  ∀n.(INTEGER(n)⊃(0+n)=n)  (2)

3  ∀x.(INDVAR(x)⊃(len(x)-1)=0)  (3)
```

4 ∀n.(INTEGER(n)⊃(n-0)=n) (4)

5 ∀x.(INDVAR(x)⊃(1 gl x)=x) (5)

6 INDVAR(x)∧(FORM(f)∧(INTEGER(n)∧SUBT(x,f,n))) (6)

7 INTEGER(1) (7)

8 SUBT(x,f,n)≡(TERM(x)∧(FORM(f)∧(INTEGER(n)∧∀x2 k.((INDVAR(x2)∧(INTEGER(k)∧(k gl x)=x2))⊃FRN(x2,n-(len(x)-k),f)))))

9 ∀x2 k.((INDVAR(x2)∧(INTEGER(k)∧(k gl x)=x2))⊃FRN(x2,n-(len(x)-k),f)) (6)

10 (INDVAR(x)∧(INTEGER(1)∧(1 gl x)=x))⊃FRN(x,n-(len(x)-1),f) (6)

11 INDVAR(x)⊃(1 gl x)=x (5)

12 (1 gl x)=x (5 6 7)

13 (INDVAR(x)∧(INTEGER(1)∧x=x))⊃FRN(x,n-(len(x)-1),f) (5 6 7)

14 INDVAR(x)⊃(len(x)-1)=0 (3)

15 (len(x)-1)=0 (3 5 6 7)

16 (INDVAR(x)∧(INTEGER(1)∧x=x))⊃FRN(x,n-0,f) (3 5 6 7)

17 INTEGER(n)⊃(n-0)=n (4)

18 (n-0)=n (3 4 5 6 7)

19 (INDVAR(x)∧(INTEGER(1)∧x=x))⊃FRN(x,n,f) (3 4 5 6 7)

20 FRN(x,n,f) (3 4 5 6 7)

21 (INDVAR(x)∧(FORM(f)∧(INTEGER(n)∧SUBT(x,f,n))))⊃FRN(x,n,f) (3 4 5 7)

22 ∀x f n.((INDVAR(x)∧(FORM(f)∧(INTEGER(n)∧SUBT(x,f,n))))⊃FRN(x,n,f)) (3 4 5 7)

23 INVART(k,f1,k,f2)≡(INTEGER(k)∧(FORM(f1)∧(INTEGER(k)∧(FORM(f2)∧((GEB(k gl f2,k,f2)≡GEB(k gl f1,k,f1))∧((FRN(k gl f2,k,f2)≡FRN(k gl f1,k,f1))∧(k gl f2)=(k gl f1)))))))

24 INVARV(k,f1,f2)≡(INTEGER(k)∧(FORM(f1)∧(FORM(f2)∧((GEB(k gl f2,k,f2)≡GEB(k gl f1,k,f1))∧((FRN(k gl f2,k,f2)≡FRN(k gl f1,k,f1))∧(k gl f2)=(k gl f1))))))

25 INVART(k,f1,k,f2)≡INVARV(k,f1,f2)

26 ∀k f1 f2.(INVART(k,f1,k,f2)≡INVARV(k,f1,f2))

27 INDVAR(x1)∧(INDVAR(x2)∧(FORM(f1)∧(FORM(f2)∧SBT(x1,x2,f1,f2)))) (27)

28 ∀x2.(INDVAR(x2)⊃TERM(x2)) (28)

29 INDVAR(x2)⊃TERM(x2) (28)

30 ∀x1 n1 f1.INTEGER(numbfreeocc(x1,n1,f1)) (30)

31 SBT(x1,x2,f1,f2)≡((INDVAR(x1)∧(TERM(x2)∧(FORM(f1)∧FORM(f2))))⊃∀n1 n2.((INTEGER(n1)∧
(INTEGER(n2)∧n2=((numbfreeocc(x1,n1,f1)*(len(x2)-1))+n1)))⊃((¬INDVAR(n1 gl f1)⊃
(n1 gl f1)=(n2 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃SUBT(x2,f2,n2))∧
(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n2,f2)))))))

32 ∀n1 n2.((INTEGER(n1)∧(INTEGER(n2)∧n2=((numbfreeocc(x1,n1,f1)*(len(x2)-1))+n1)))⊃
((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n2 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃
SUBT(x2,f2,n2))∧(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n2,f2)))))) (27 28 30)

33 (INTEGER(n1)∧(INTEGER(n1)∧n1=((numbfreeocc(x1,n1,f1)*(len(x2)-1))+n1)))⊃(
(¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃
SUBT(x2,f2,n1))∧(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n1,f2))))) (27 28 30)

34 INTEGER(n1) (34)

35 INTEGER(numbfreeocc(x1,n1,f1)) (30)

36 (INTEGER(numbfreeocc(x1,n1,f1))∧INDVAR(x2))⊃(numbfreeocc(x1,n1,f1)*(len(x2)-1))=0 (1)

37 (numbfreeocc(x1,n1,f1)*(len(x2)-1))=0 (1 27 28 30 34)

38 (INTEGER(n1)∧(INTEGER(n1)∧n1=(0+n1)))⊃((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧
(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃SUBT(x2,f2,n1))∧(¬FRN(x1,n1,f1)⊃
INVART(n1,f1,n1,f2))))) (1 27 28 30 34)

39 INTEGER(n1)⊃(0+n1)=n1 (2)

40 (0+n1)=n1 (1 2 27 28 30 34)

41 (INTEGER(n1)∧(INTEGER(n1)∧n1=n1))⊃((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧
(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃SUBT(x2,f2,n1))∧(¬FRN(x1,n1,f1)⊃
INVART(n1,f1,n1,f2))))) (1 2 27 28 30 34)

42 (¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃
SUBT(x2,f2,n1))∧(¬FRN(x1,n1,f1)⊃INVART(n1,f1,n1,f2)))) (1 2 27 28 30 34)

43 SBV(x1,x2,f1,f2)≡((INDVAR(x1)∧(INDVAR(x2)∧(FORM(f1)∧FORM(f2))))⊃∀n.(INTEGER(n)⊃(
(¬INDVAR(n gl f1)⊃(n gl f1)=(n gl f2))∧(INDVAR(n gl f1)⊃((FRN(x1,n,f1)⊃
FRN(x2,n,f2))∧(¬FRN(x1,n,f1)⊃INVARV(n,f1,f2)))))))

44 (INDVAR(x2)∧(FORM(f2)∧(INTEGER(n1)∧SUBT(x2,f2,n1))))⊃FRN(x2,n1,f2) (3 4 5 7)

45 INVART(n1,f1,n1,f2)≡INVARV(n1,f1,f2)

46 (¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃
FRN(x2,n1,f2))∧(¬FRN(x1,n1,f1)⊃INVARV(n1,f1,f2)))) (1 2 3 4 5 7 27 28 30 34)

47 INTEGER(n1)⊃((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃
(FRN(x1,n1,f1)⊃FRN(x2,n1,f2))∧(¬FRN(x1,n1,f1)⊃INVARV(n1,f1,f2))))) (1 2 3 4 5

7 27 28 30)

48 ∀n1.(INTEGER(n1)⊃((¬INDVAR(n1 gl f1)⊃(n1 gl f1)=(n1 gl f2))∧(INDVAR(n1 gl f1)⊃((FRN(x1,n1,f1)⊃FRN(x2,n1,f2))∧(¬FRN(x1,n1,f1)⊃INVARV(n1,f1,f2)))))) (1 2 3 4 5 7 27 28 30)

49 SBV(x1,x2,f1,f2) (1 2 3 4 5 7 27 28 30 34)

50 (INDVAR(x1)∧(INDVAR(x2)∧(FORM(f1)∧(FORM(f2)∧SBT(x1,x2,f1,f2)))))⊃SBV(x1,x2,f1,f2) (1 2 3 4 5 7 28 30 34)

51 ∀x1 x2 f1 f2.((INDVAR(x1)∧(INDVAR(x2)∧(FORM(f1)∧(FORM(f2)∧SBT(x1,x2,f1,f2)))))⊃SBV(x1,x2,f1,f2)) (1 2 3 4 5 7 28 30)

APPENDIX 5

THE PROOF THAT UNIVERSAL QUANTIFIER CAN BE INTERCHANGED

5.1 FOL commands for the main lemma in the many sorted logic

```
LABEL TH1; ASSUME ∀x1 x2 f1 f2.(SBT(x1,x2,f1,f2)⊃ SBV(x1,x2,f1,f2));
∀e TH1, x,x,f1,sbt(x,x,f1);
∀E SUBSTDF3 x,x, f1, sbt(x,x,f1);
∀e SUBDEFC x, x,f1,sbt(x,x,f1);
tauteq -:#2,1:-;
∀e -,n;
∀E FREEVO, x, n, f1;
∀E FREEVO, x, n, sbt(x,x,f1);
∀E SUBDEF1 n, f1, sbt(x,x,f1);
tauteq (n gl f1)=(n gl sbt(x,x,f1)),11, 17, 18;
∀i -,n;
∀E EQS f1,sbt(x,x,f1);
tauteq sbt(x,x,f1)=f1,--,-;
∀i -,x,f1←f;
```

5.2 Printout of the proof in the many sorted logic

```
1 ∀x1 x2 f1 f2.(SBT(x1,x2,f1,f2)⊃SBV(x1,x2,f1,f2)) (1)

2 SBT(x,x,f1,sbt(x,x,f1))⊃SBV(x,x,f1,sbt(x,x,f1)) (1)

3 SBT(x,x,f1,sbt(x,x,f1))≡sbt(x,x,f1)=sbt(x,x,f1)

4 SBV(x,x,f1,sbt(x,x,f1))≡∀n.((¬INDVAR(n gl f1)⊃(n gl f1)=(n gl sbt(x,x,f1)))∧
  (INDVAR(n gl f1)⊃((FRN(x,n,f1)⊃FRN(x,n,sbt(x,x,f1)))∧(¬FRN(x,n,f1)⊃INVARV(n,
  f1,sbt(x,x,f1))))))

5 ∀n.((¬INDVAR(n gl f1)⊃(n gl f1)=(n gl sbt(x,x,f1)))∧(INDVAR(n gl f1)⊃((FRN(x,
  n,f1)⊃FRN(x,n,sbt(x,x,f1)))∧(¬FRN(x,n,f1)⊃INVARV(n,f1,sbt(x,x,f1)))))) (1)

6 (¬INDVAR(n gl f1)⊃(n gl f1)=(n gl sbt(x,x,f1)))∧(INDVAR(n gl f1)⊃((FRN(x,n,f1)⊃
  FRN(x,n,sbt(x,x,f1)))∧(¬FRN(x,n,f1)⊃INVARV(n,f1,sbt(x,x,f1))))) (1)

7 FRN(x,n,f1)≡(x=(n gl f1)∧¬GEB(x,n,f1))    ∀E FREEVO x , n , f1

8 FRN(x,n,sbt(x,x,f1))≡(x=(n gl sbt(x,x,f1))∧¬GEB(x,n,sbt(x,x,f1)))

9 INVARV(n,f1,sbt(x,x,f1))≡((GEB(n gl sbt(x,x,f1),n,sbt(x,x,f1))≡GEB(n gl f1,n,f1))∧
  ((FRN(n gl sbt(x,x,f1),n,sbt(x,x,f1))≡FRN(n gl f1,n,f1))∧(n gl sbt(x,x,f1))=
  (n gl f1)))

10 (n gl f1)=(n gl sbt(x,x,f1)) (1)
```

```
11  ∀n.(n gl f1)=(n gl sbt(x,x,f1))  (1)

12  ∀n.(n gl f1)=(n gl sbt(x,x,f1))=f1=sbt(x,x,f1)

13  sbt(x,x,f1)= f1  (1)

14  ∀x f.sbt(x,x,f)=f  (1)
```

5.3 FOL commands for the theorem in the many sorted logic

```
LABEL FIRSTLEMMA;
ASSUME ∀x f.sbt(x,x,f) =f;

LABEL THEON1;
ASSUME ∀f sq.scar(f cc sq) = f ;
LABEL THEON2;
ASSUME ∀f sq.scdr(f cc sq) = sq;
```

Proof of the Lemma: BEW(x gen f) ⊃ BEW(f)

```
LABEL HPT;
ASSUME BEW(x gen f)  ;

LABEL THTAUT;
∀e FIRSTLEMMA   x, f;

∀e PROVABLE x gen f ;
TAUT -:#2 , -,HPT;
LABEL HPAUX;
∃e - ,sq ;


∀e GENRULO f cc sq ,sq,x,x;
LABEL THN1;
∀e THEON1    f, sq;
∀e THEON2    f, sq;
TAUTEQ    ---:#2#2#2#1[f1← f] ,1:-;
UNIFY    ----:#2#2#2  , -;
TAUTEQ  -----:#1     , 1:-;


∀e PROOF  f cc sq ;
LABEL GENE1;
∨I  GEN1(f cc sq,sq,x,x) , -- , EXI(f cc sq,sq,x,x) ;
UNIFY  --:#2#2#2#1 , - ;
LABEL PROOFTR;
TAUT  ---:#1, 1:-;

∧e HPAUX :#2#2;
∀e  -    , f1;
```

```
∀e DEPENDO f cc sq, sq,f1;
UNIFY -:*2*2*2*2*2, GENE1 ;

TAUTEQ DEPEND(f cc sq,f1) ⊃ AXIOM (f1) ,1:-;
∀i -,f1←f1;
TAUTEQ THN1:*2 ≡ THN1:*1 ,THN1;
∧i PROOFTR, - , -- ;
LABEL USEFUL;
∀e PROVABLE f;
UNIFY -:*2 ,--;
TAUT --:*1,1:-;
LABEL C1TH1;
⊃I HPT,-;
```

Proof of the Lemma: BEW(f) ⊃ BEW(x gen f)

```
LABEL HPT1;
ASSUME BEW(f);

TAUT USEFUL:*2 , -,HPT1,USEFUL;
∃e - ,sq;

∧e -:*2*2;
∀e - , f1;
∀e GENRUL2 x,sq;
∀e THEORY x,f1;
TAUTEQ --:*2*1*1[f←f1] ,HPT1:-;
∀i     - ,f1←f1;
TAUT ----:*1 ,HPT1:-;

∀e GENRUL1 ((x gen f) cc sq) , sq ,x,x ;
LABEL THN2;
∀e THEON1    x gen f , sq ;
∀e THEON2    x gen f , sq ;
TAUTEQ    ---:*2*2*2*1[f1 ← f] , THTAUT,HPT1:-;
UNIFY    ----:*2*2*2  , -;
TAUTEQ  -----:*1      , THTAUT,HPT1:-;


∀e PROOF  (x gen f) cc sq ;
LABEL GEN1;
∨i -- , GENE((x gen f) cc sq,sq,x,x) , EXI((x gen f) cc sq,sq,x,x) ;
UNIFY  --:*2*2*2*1 , - ;

LABEL PROOFTR1;
TAUT  ---:*1, HPT1:-,THTAUT;


∀e DEPENDO (x gen f) cc sq, sq,f1;
∃I GEN1  ,x←f OCC 3 6 9,x←x1 OCC 2 4 6;


TAUTEQ DEPEND((x gen f) cc sq,f1) ⊃ AXIOM (f1) ,THTAUT,HPT1:-;
```

```
∀i -,f1←f1;
TAUTEQ THN2:#2 ≡ THN2:#1 ,THN2;
∧i PROOFTR1, - , -- ;
∀e PROVABLE x gen f;
UNIFY -:#2 ,--;
TAUT --:#1,THTAUT,HPT1:-;
LABEL C2TH1;
⊃I HPT1,-;
≡I C1TH1,C2TH1;
LABEL TH1;
∀I -,x,f;
∀e TH1 x1,x2 gen f;
∀e TH1 x2,f;
∀e TH1 x1,f;
∀e TH1 x2,x1 gen f;
TAUT ----:#1 ⊃ -:#1, TH1:-;
∀i -,x1,x2,f;
```

5.4 Printout of the proof of the theorem in the many sorted logic

```
1  ∀x f.sbt(x,x,f)=f  (1)

2  ∀f sq.scar(f cc sq)=f  (2)

3  ∀f sq.scdr(f cc sq)=sq  (3)

4  BEW(x gen f)  (4)

5  sbt(x,x,f)=f  (1)

6  BEW(x gen f)≡∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))

7  ∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))  (4)

8  PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)))  (8)

9  GENE(f cc sq,sq,x,x)≡(scdr(f cc sq)=sq∧(PROOFTREE(sq)∧∃f1.(scar(sq)=(x gen f1)∧
   scar(f cc sq)=sbt(x,x,f1))))

10  scar(f cc sq)=f  (2)

11  scdr(f cc sq)=sq  (3)

12  scar(sq)=(x gen f)∧scar(f cc sq)=sbt(x,x,f)  (1 2 3 4 8)

13  ∃f1.(scar(sq)=(x gen f1)∧scar(f cc sq)=sbt(x,x,f1))  (1 2 3 4 8)

14  GENE(f cc sq,sq,x,x)  (1 2 3 4 8)

15  PROOFTREE(f cc sq)≡(FORM(f cc sq)∨(∃pf.(ORI(f cc sq,pf)∨(ANDE(f cc sq,pf)∨
    (FALSEE(f cc sq,pf)∨(NOTI(f cc sq,pf)∨(NOTE(f cc sq,pf)∨IMPLI(f cc sq,pf))))))∨
```

(∃pf x t.(GENI(f cc sq,pf,x,f)∨(GENE(f cc sq,pf,x,f)∨EXI(f cc sq,pf,x,t)))∨
(∃pf1 pf2.(ANDI(f cc sq,pf1,pf2)∨(FALSEI(f cc sq,pf1,pf2)∨IMPLE(f cc sq,pf1,pf2)))∨
∃pf1 pf2 x f.EXE(f cc sq,pf1,pf2,x,f)∨∃pf1 pf2 pf3.ORE(f cc sq,pf1,pf2,pf3))))))

16 GENI(f cc sq,sq,x,x)∨(GENE(f cc sq,sq,x,x)∨EXI(f cc sq,sq,x,x)) (1 2 3 4 8)

17 ∃pf x t.(GENI(f cc sq,pf,x,f)∨(GENE(f cc sq,pf,x,t)∨EXI(f cc sq,pf,x,t))) (1 2 3 4 8)

18 PROOFTREE(f cc sq) (1 2 3 4 8)

19 ∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)) (8)

20 DEPEND(sq,f1)⊃AXIOM(f1) (8)

21 PROOFTREE(f cc sq)⊃(PROOFTREE(sq)⊃((sq=scdr(f cc sq)⊃(DEPEND(f cc sq,f1)≡
DEPEND(sq,f1)))≡(ORI(f cc sq,sq)∨(ANDE(f cc sq,sq)∨(FALSEE(f cc sq,sq)∨
(∃f.((NOTID(f cc sq,sq,f)∨(NOTED(f cc sq,sq,f)∨IMPLID(f cc sq,sq,f)))∧f≠f1)∨
∃x t.(GENI(f cc sq,sq,x,t)∨(GENE(f cc sq,sq,x,t)∨EXI(f cc sq,sq,x,t)))))))))

22 ∃x t.(GENI(f cc sq,sq,x,t)∨(GENE(f cc sq,sq,x,t)∨EXI(f cc sq,sq,x,t))) (1 2 3 4 8)

23 DEPEND(f cc sq,f1)⊃AXIOM(f1) (1 2 3 4 8)

24 ∀f1.(DEPEND(f cc sq,f1)⊃AXIOM(f1)) (1 2 3 4 8)

25 f=scar(f cc sq) (2)

26 PROOFTREE(f cc sq)∧(f=scar(f cc sq)∧∀f1.(DEPEND(f cc sq,f1)⊃AXIOM(f1))) (1 2 3 4 8)

27 BEW(f)≡∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))

28 ∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)))) (1 2 3 4)

29 BEW(f) (1 2 3 4)

30 BEW(x gen f)⊃BEW(f) (1 2 3)

31 BEW(f) (31)

32 ∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)))) (31)

33 PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))) (33)

34 ∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)) (33)

35 DEPEND(sq,f1)⊃AXIOM(f1) (33)

36 APGENI(x,sq)≡(∀f.(DEPEND(sq,f)⊃¬FR(x,f))∧PROOFTREE(sq))

37 AXIOM(f1)⊃¬FR(x,f1)

38 DEPEND(sq,f1)⊃¬FR(x,f1) (31 33)

39 ∀f1.(DEPEND(sq,f1)⊃¬FR(x,f1)) (31 33)

40 APGENI(x,sq) (31 33)

41 GENI((x gen f) cc sq,sq,x,x)≡(scdr((x gen f) cc sq)=sq∧(PROOFTREE(sq)∧
∃f1.(scar((x gen f) cc sq)=(x gen f1)∧(scar(sq)=sbl(x,x,f1)∧APGENI(x,sq)))))

42 scar((x gen f) cc sq)=(x gen f) (2)

43 scdr((x gen f) cc sq)=sq (3)

44 scar((x gen f) cc sq)=(x gen f)∧(scar(sq)=sbt(x,x,f)∧APGENI(x,sq)) (1 2 3 31 33)

45 ∃f1.(scar((x gen f) cc sq)=(x gen f1)∧(scar(sq)=sbt(x,x,f1)∧APGENI(x,sq))) (1 2 3 31 33)

46 GENI((x gen f) cc sq,sq,x,x) (1 2 3 31 33)

47 PROOFTREE((x gen f) cc sq)≡(FORM((x gen f) cc sq)∨(∃pf.(ORI((x gen f) cc sq,pf)∨
(ANDE((x gen f) cc sq,pf)∨(FALSEE((x gen f) cc sq,pf)∨(NOTI((x gen f) cc sq,pf)∨
(NOTE((x gen f) cc sq,pf)∨IMPLI((x gen f) cc sq,pf))))))∨(∃pf x1 f.(GENI((x gen f)
cc sq,pf,x1,t)∨(GENE((x gen f) cc sq,pf,x1,t)∨EXI((x gen f) cc sq,pf,x1,t)))∨
(∃pf1 pf2.(ANDI((x gen f) cc sq,pf1,pf2)∨(FALSEI((x gen f) cc sq,pf1,pf2)∨
IMPLE((x gen f) cc sq,pf1,pf2)))∨(∃pf1 pf2 x1 f.EXE((x gen f) cc sq,pf1,pf2,x1,t)∨
∃pf1 pf2 pf3.ORE((x gen f) cc sq,pf1,pf2,pf3))))))

48 GENI((x gen f) cc sq,sq,x,x)∨(GENE((x gen f) cc sq,sq,x,x)∨EXI((x gen f) cc sq,
sq,x,x)) (1 2 3 31 33)

49 ∃pf x1 f.(GENI((x gen f) cc sq,pf,x1,t)∨(GENE((x gen f) cc sq,pf,x1,f)∨
EXI((x gen f) cc sq,pf,x1,t))) (1 2 3 31 33)

50 PROOFTREE((x gen f) cc sq) (1 2 3 31 33)

51 PROOFTREE((x gen f) cc sq)⊃(PROOFTREE(sq)⊃((sq=scdr((x gen f) cc sq)⊃(DEPEND(
(x gen f) cc sq,f1)≡DEPEND(sq,f1)))≡(ORI((x gen f) cc sq,sq)∨(ANDE((x gen f)
cc sq,sq)∨(FALSEE((x gen f) cc sq,sq)∨(∃f.((NOTID((x gen f) cc sq,sq,f)∨
(NOTED((x gen f) cc sq,sq,f)∨IMPLID((x gen f) cc sq,sq,f)))∧f≠f1)∨
∃x1 f.(GENI((x gen f) cc sq,sq,x1,t)∨(GENE((x gen f) cc sq,sq,x1,t)∨EXI(
(x gen f) cc sq,sq,x1,f)))))))))

52 ∃x1 t.(GENI((x gen f) cc sq,sq,x1,t)∨(GENE((x gen f) cc sq,sq,x1,f)∨
EXI((x gen f) cc sq,sq,x1,f))) (1 2 3 31 33)

53 DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1) (1 2 3 31 33)

54 ∀f1.(DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1)) (1 2 3 31 33)

55 (x gen f)=scar((x gen f) cc sq) (2)

56 PROOFTREE((x gen f) cc sq)∧((x gen f)=scar((x gen f) cc sq)∧
∀f1.(DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1))) (1 2 3 31 33)

57 BEW(x gen f)≡∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))

```
58 ∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))  (1 2 3 31)

59 BEW(x gen f)  (1 2 3 31)

60 BEW(f)⊃BEW(x gen f)  (1 2 3)

61 BEW(x gen f)≡BEW(f)  (1 2 3)

62 ∀x f.(BEW(x gen f)≡BEW(f))  (1 2 3)

63 BEW(x1 gen (x2 gen f))≡BEW(x2 gen f)  (1 2 3)

64 BEW(x2 gen f)≡BEW(f)  (1 2 3)

65 BEW(x1 gen f)≡BEW(f)  (1 2 3)

66 BEW(x2 gen (x1 gen f))≡BEW(x1 gen f)  (1 2 3)

67 BEW(x1 gen (x2 gen f))⊃BEW(x2 gen (x1 gen f))  (1 2 3)

68 ∀x1 x2 f. (BEW(x1 gen (x2 gen f)) ⊃ BEW(x2 gen (x1 gen f))) (1 2 3)
```

5.5 FOL commands for the main lemma in the earlier axiomatization

```
LABEL HPT;  ASSUME INDVAR(x) ∧ FORM(f1) ;
LABEL TH1 ; ASSUME ∀x1 x2 f1 f2.((INDVAR(x1) ∧ INDVAR(x2) ∧ FORM(f1) ∧ FORM(f2) ∧
                   SBT(x1,x2,f1,f2)) ⊃ SBV(x1,x2,f1,f2));
LABEL TH2 ; ASSUME ∀x.(INDVAR(x) ⊃ TERM(x));
LABEL TH3 ; ASSUME ∀x.(FORM(x) ⊃ STRING(x));
∀e TH1, x,x,f1,sbt(x,x,f1);
∀e TH2, x;
∀e TH3, f1;
∀e TH3, sbt(x,x,f1);
∀E SUBSTDF3 x,x, f1, sbt(x,x,f1);
∀E SUBSTDF4 x,x, f1;
∀e SUBDEF0 x, x,f1,sbt(x,x,f1);
tauteq -:#2#2,1:-;
∀e -,n;
∀E FREEV0, x, n, f1;
∀E FREEV0, x, n, sbt(x,x,f1);
∀E SUBDEF1  n, f1, sbt(x,x,f1);
tauteq INTEGER(n) ⊃ ((n gl f1)=(n gl sbt(x,x,f1))) 1:-;
∀i -,n;
∀E EQS, f1,sbt(x,x,f1);
taut -:#2#2,1:-;
⊃i,1:-;
∀I -,x,f1←f;
```

5.6 Printout of the proof of the main lemma in the second axiomatization

```
1  INDVAR(x)∧FORM(f1)  (1)  ASSUME

2  ∀x1 x2 f1 f2.((INDVAR(x1)∧(INDVAR(x2)∧(FORM(f1)∧(FORM(f2)∧SBT(x1,x2,f1,f2)))))⊃
   SBV(x1,x2,f1,f2))  (2)  ASSUME

3  ∀x.(INDVAR(x)⊃TERM(x))  (3)  ASSUME

4  ∀x.(FORM(x)⊃STRING(x))  (4)  ASSUME

5  (INDVAR(x)∧(INDVAR(x)∧(FORM(f1)∧(FORM(sbt(x,x,f1))∧SBT(x,x,f1,sbt(x,x,f1))))))⊃
   SBV(x,x,f1,sbt(x,x,f1))  (2)  ∀E 2 x , x , f1 , sbt(x,x,f1)

6  INDVAR(x)⊃TERM(x)  (3)  ∀E 3 x

7  FORM(f1)⊃STRING(f1)  (4)  ∀E 4 f1

8  FORM(sbt(x,x,f1))⊃STRING(sbt(x,x,f1))  (4)  ∀E 4 sbt(x,x,f1)

9  (INDVAR(x)∧(TERM(x)∧(FORM(f1)∧FORM(sbt(x,x,f1)))))⊃(SBT(x,x,f1,sbt(x,x,f1))≡
   sbt(x,x,f1)=sbt(x,x,f1))    ∀E SUBSTDF3 x , x , f1 , sbt(x,x,f1)

10  (INDVAR(x)∧(TERM(x)∧FORM(f1)))⊃FORM(sbt(x,x,f1))    ∀E SUBSTDF4 x , x , f1

11  SBV(x,x,f1,sbt(x,x,f1)) ((INDVAR(x)∧(INDVAR(x)∧(FORM(f1)∧FORM(sbt(x,x,f1)))))⊃
   ∀n.(INTEGER(n)⊃((¬INDVAR(n gl f1)⊃(n gl f1)=(n gl sbt(x,x,f1)))∧(INDVAR(n gl f1)
   ⊃((FRN(x,n,f1)⊃FRN(x,n,sbt(x,x,f1)))∧(¬FRN(x,n,f1)⊃INVARV(n,f1 ,sbt(x,x,f1)))))))
   ))   ∀E SUBDEF0 x , x , f1 , sbt(x,x,f1)

12  ∀n.(INTEGER(n)⊃((¬INDVAR(n gl f1)⊃(n gl f1)=(n gl sbt(x,x,f1)))∧(INDVAR(n gl f1)
   ⊃((FRN(x,n,f1)⊃FRN(x,n,sbt(x ,x,f1)))∧(¬FRN(x,n,f1)⊃
   INVARV(n,f1,sbt(x,x,f1)))))))  (1 2 3 4)  1 : 11

13  INTEGER(n)⊃((¬INDVAR(n gl f1)⊃(n gl f1)=(n gl sbt(x,x,f1)))∧(INDVAR(n gl f1)⊃
   ((FRN(x,n,f1)⊃FRN(x,n,sbt(x,x,f1)))∧(¬FRN(x,n,f1)⊃INVARV(n,f1,sbt(x,x,f1))))))
   (1 2 3 4)  ∀E 12 n

14  FRN(x,n,f1)≡(x=(n gl f1)∧¬GEB(x,n,f1))    ∀E FREEV0 x , n , f1

15  FRN(x,n,sbt(x,x,f1))≡(x=(n gl sbt(x,x,f1))∧¬GEB(x,n,sbt(x,x,f1)))    ∀E FREEV0 x , n , sbt(x,x,f1)

16  INVARV(n,f1,sbt(x,x,f1))≡(INTEGER(n)∧(FORM(f1)∧(FORM(sbt(x,x,f1))∧((GEB(n gl
   sbt(x,x,f1),n,sbt(x,x,f1))≡GEB(
   n gl f1,n,f1))∧((FRN(n gl sbt(x,x,f1),n,sbt(x,x,f1))≡FRN(n gl f1,n,f1))∧(n gl
   sbt (x,x,f1))=(n gl f1))))))    ∀E  SUBDEF1 n , f1 , sbt(x,x,f1)

17  INTEGER(n)⊃(n gl f1)=(n gl sbt(x,x,f1))  (1 2 3 4)  1 : 16

18  ∀n.(INTEGER(n)⊃(n gl f1)=(n gl sbt(x,x,f1)))  (1 2 3 4)  ∀I 17 n ← n

19  (STRING(f1)∧STRING(sbt(x,x,f1)))⊃(∀n.(INTEGER(n)⊃(n gl f1)=(n gl sbt(x,x,f1)))≡
   f1=sbt(x,x,f1))    ∀E EQS f1 , sbt(x,x,f1)
```

```
20  f1=sbt(x,x,f1)  (1 2 3 4)  1 : 19

21  (INDVAR(x)∧FORM(f1))⊃f1=sbt(x,x,f1)  (2 3 4)  ⊃I 1 20

22  ∀x f.((INDVAR(x)∧FORM(f))⊃f=sbt(x,x,f))  (2 3 4)  ∀I 21 x ← f f1 ← x
```

5.7 FOL commands in the earlier axiomatization

```
LABEL FIRSTLEMMA;
ASSUME ∀x f.((INDVAR(x)∧ FORM(f)) ⊃ sbt(x,x,f) =f);

LABEL THEON1;
ASSUME ∀s sq.((STRING(s)∧ SEQUENCE(sq))⊃ scar(s cc sq) = s);
LABEL THEON2;
ASSUME ∀s sq.((STRING(s)∧ SEQUENCE(sq))⊃scdr(s cc sq) = sq);
LABEL TH1;
ASSUME ∀x f.((INDVAR(x)∧FORM(f))⊃FORM(x gen f));
LABEL TH2;
ASSUME ∀f.(FORM(f) ⊃ STRING(f)) ;
LABEL TH3;
ASSUME ∀f sq.((FORM(f)∧SEQUENCE(sq))⊃SEQUENCE(f cc sq));
LABEL TH4;
ASSUME ∀x.(INDVAR(x)⊃ TERM(x));
LABEL TH5;
ASSUME ∀pf.(PROOFTREE(pf)⊃SEQUENCE(pf));
```

Proof of the Lemma BEW(x gen f) ⊃ BEW(f) Under the Assumption: INDVAR(x) ∧ FORM(f)

```
LABEL HPTT;
ASSUME INDVAR(x)∧ FORM(f);
LABEL HPT;
ASSUME BEW(x gen f)  ;

LABEL THTAUT;
∀e FIRSTLEMMA  x, f;

∀e PROVABLE x gen f ;
∀E TH1 x,f;
TAUT --:*2*2, HPTT:-;
∀e TH2,f;
∀e TH3,f,sq;
∀E TH4,x;
∀E TH5,sq;
LABEL HPAUX;
∃e ----- ,sq ;


∀e GENRULO f cc sq ,sq,x,x;
LABEL THN1;
∀e THEON1    f, sq;
∀e THEON2    f, sq;
TAUTEQ    ---:*2*2*2*2*2*2*1[f1← f] ,1:-;
```

```
UNIFY   ----:*2*2*2*2*2*2 , -;
TAUTEQ -----:*1     , 1:-;

∀e PROOF  f cc sq ;
LABEL GENE1;
TAUTEQ PROOFTREE(sq)∧INDVAR(x)∧TERM(x)∧(GENI(f cc sq,sq,x,x) ∨ --: ∨
      EXI(f cc sq,sq,x,x)) 1:-;
UNIFY --:*2*2*2*1 , - ;
LABEL PROOFTR;
TAUT ---:*1, 1:-;

∧e HPAUX :*2*2;
∀e  -   , f1;

∀e DEPEND f cc sq, sq,f1;
∧E GENE1:*2;
UNIFY --:*2*2*2*2*2, - ;

TAUTEQ DEPEND(f cc sq,f1) ⊃ AXIOM (f1) ,1:-;
∀i -,f1←f1;
TAUTEQ f=scar(f cc sq) 1:-;
∧i PROOFTR, - , -- ;
LABEL USEFUL;
∀e PROVABLE f;
UNIFY -:*2*2 ,--;
TAUT --:*1,1:-;
LABEL CITH1;
⊃I HPT,-;
```

Proof of the Lemma BEW(f) ⊃ BEW(x gen f) Under the Assumption: INDVAR(x) ∧ FORM(f)

```
LABEL HPT1;
ASSUME BEW(f);

TAUT USEFUL:*2 , -,HPT1,USEFUL;
∧E -:*2
∃e - ,sq;

∧e -:*2*2;
∀e - , f1;
∀e GENRUL2 x,sq;
∀e THEORY x,f1;
TAUTEQ --:*2*1*2*1[f←f1] ,HPTT,HPT1:-;
∀i     - ,f1←f1;
TAUT ----:*1 ,HPTT,HPT1:-;

∀e GENRUL1 ((x gen f) cc sq) , sq ,x,x ;
LABEL THN2;
∀e THEON1    x gen f , sq ;
∀e THEON2    x gen f , sq ;
∀E TH1 x ,f;
∀E TH2 x gen f;
∀E TH5 sq;
TAUTEQ  ------:*2*2*2*1[f1 ← f] ,HPTT, THTAUT,HPT1:-;
```

```
UNIFY --------:*2*2*2 , -;
∀E TH3, x gen f,sq;
TAUTEQ ---------:*1, HPTT, THTAUT,HPT1:-;



∀e PROOF (x gen f) cc sq .
∀e TH4,x;
LABEL GEN1;
TAUTEQ PROOFTREE(sq) ∧ INDVAR(x) ∧ TERM(x) ∧ ( ---: ∨ GENE((x gen f) cc sq,sq,x,x) ∨
EXI((x gen f) cc sq,sq,x,x)) HPTT,HPT1:-;
UNIFY ---:*2*2*2*1 , - ;

LABEL PROOFTR1;
TAUT ----:*1, HPT1:-,THTAUT,HPTT;

∀e DEPEND (x gen f) cc sq, sq,f1;
∧E GEN1:*2;
∃i - ,x←t OCC 2 5 8 11;
∃i -, x←x1 OCC 1 3 5 7;

TAUTEQ DEPEND((x gen f) cc sq,f1) ⊃ AXIOM (f1) ,THTAUT,HPTT,HPT1:-;
∀i -,f1←f1;
TAUTEQ x gen f = scar((x gen f) cc sq),HPTT,HPT1:-;
∧i PROOFTR1, - , -- ;
∀e PROVABLE x gen f;
UNIFY -:*2*2 ,--;
TAUT --:*1,THTAUT,HPT1:-;
LABEL C2TH1;
⊃I HPT1,-;
≡I C1TH1,C2TH1;
LABEL THGEN;
⊃I HPTT,-;
∀I -,x,f;
∀e TH1 x1,x2 gen f;
∀e TH1 x2,f;
∀e TH1 x1,f;
∀e TH1 x2,x1 gen f;
∀E TH1,x1,f;
∀E TH1,x2,f;
TAUT (INDVAR(x1) ∧ (INDVAR(x2) ∧ FORM(f))) ⊃ (BEW(x1 gen (x2 gen f)) ≡
BEW(x2 gen (x1 gen f))),THGEN:-;
∀I -,x1,x2,f;
```

5.6 Printout of the proof in the earlier axiomatization

```
1  ∀x f.((INDVAR(x)∧FORM(f))⊃sbt(x,x,f)=f)  (1)  ASSUME

2  ∀s sq.((STRING(s)∧SEQUENCE(sq))⊃scar(s cc sq)=s)  (2)  ASSUME

3  ∀s sq.((STRING(s)∧SEQUENCE(sq))⊃scdr(s cc sq)=sq)  (3)  ASSUME
```

4 ∀x f.((INDVAR(x)∧FORM(f))⊃FORM(x gen f)) (4) ASSUME

5 ∀f.(FORM(f)⊃STRING(f)) (5) ASSUME

6 ∀f sq.((FORM(f)∧SEQUENCE(sq))⊃SEQUENCE(f cc sq)) (6) ASSUME

7 ∀x.(INDVAR(x)⊃TERM(x)) (7) ASSUME

8 ∀pf.(PROOFTREE(pf)⊃SEQUENCE(pf)) (8) ASSUME

9 INDVAR(x)∧FORM(f) (9) ASSUME

10 BEW(x gen f) (10) ASSUME

11 (INDVAR(x)∧FORM(f))⊃sbt(x,x,f)=f (1) ∀E 1 x , f

12 BEW(x gen f)≡(FORM(x gen f)∧∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(
sq,f1)⊃AXIOM(f1))))) ∀E PROVABLE x gen f

13 (INDVAR(x)∧FORM(f))⊃FORM(x gen f) (4) ∀E 4 x , f

14 ∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)))) (1 4 9 10) 9 : 13

15 FORM(f)⊃STRING(f) (5) ∀E 5 f

16 (FORM(f)∧SEQUENCE(sq))⊃SEQUENCE(f cc sq) (6) ∀E 6 f , sq

17 INDVAR(x)⊃TERM(x) (7) ∀E 7 x

18 PROOFTREE(sq)⊃SEQUENCE(sq) (8) ∀E 8 sq

19 PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))) (19) ASSUME

20 GENE(f cc sq,sq,x,x)≡(SEQUENCE(f cc sq)∧(INDVAR(x)∧(TERM(x)∧(scdr(f cc sq)=sq∧(PROOFTREE(sq)∧
∃f1.(FORM(f1)∧(scar(sq)=(x gen f1)∧
scar(f cc sq)=sbt(x,x,f1)))))))) ∀E GENRULO f cc sq , sq , x , x

21 (STRING(f)∧SEQUENCE(sq))⊃scar(f cc sq)=f (2) ∀E 2 f , sq

22 (STRING(f)∧SEQUENCE(sq))⊃scdr(f cc sq)=sq (3) ∀E 3 f , sq

23 FORM(f)∧(scar(sq)=(x gen f)∧scar(f cc sq)=sbt(x,x,f)) (1 2 3 4 5 6 7 8 9 10 19) 1 : 22

24 ∃f1.(FORM(f1)∧(scar(sq)=(x gen f1)∧scar(f cc sq)=sbt(x,x,f1))) (1 2 3 4 5 6 7 8 9 10 19) UNIFY 23

25 GENE(f cc sq,sq,x,x) (1 2 3 4 5 6 7 8 9 10 19) 1 : 24

26 PROOFTREE(f cc sq)≡((SEQUENCE(f cc sq)∧FORM(f cc sq))∨(∃pf.(PROOFTREE(pf)∧(ORI(f cc sq,pf)∨
(ANDE(f cc sq,pf)∨(FALSEE(f cc sq,pf)∨(NOTI(f cc sq,pf)∨(NOTE(f cc sq,pf)∨
IMPLI(f cc sq,pf)))))))∨(∃pf x t.(PROOFTREE(pf)∧(INDVAR(x)∧(TERM(t)∧
(GENI(f cc sq,pf,x,t)∨(GENE(f cc sq,pf,x,t)∨
EXI(f cc sq,pf,x,t))))))∨(∃pf1 pf2.(PROOFTREE(pf1)∧(PROOFTREE(pf2)∧(ANDI(f cc sq,pf1,pf2)∨

```
(FALSEI(f cc sq,pf1,pf2)∨IMPLE(f cc sq,pf1,pf2)))))∨(∃pf1 pf2 x1 x2.(PROOFTREE(pf1)∧
(PROOFTREE(pf2)∧(INDVAR(x1)∧(INDVAR(x2)∧EXE(f cc sq,pf1,pf2,x1,x2)))))∨∃pf1 pf2 pf3.
(PROOFTREE(pf1)∧(PROOFTREE(pf2)∧(PROOFTREE(pf3)∧ORE(f cc sq,pf1,pf2,pf3)))))))))
∀E PROOF f cc sq

27  PROOFTREE(sq)∧(INDVAR(x)∧(TERM(x)∧(GENI(f cc sq,sq,x,x)∨(GENE(f cc sq,sq,x,x)∨
  EXI(f cc sq,sq,x,x)))))  (1 2 3 4 5 6 7 8 9 10 19)  1 : 26

28  ∃pf x t.(PROOFTREE(pf)∧(INDVAR(x)∧(TERM(t)∧(GENI(f cc sq,pf,x,t)∨(GENE(f cc sq,pf,x,t)∨
  EXI(f cc sq,pf,x,t))))))  (1 2 3 4 5 6 7 8 9 10 19)  UNIFY 27

29  PROOFTREE(f cc sq)  (1 2 3 4 5 6 7 8 9 10 19)  1 : 28

30  ∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))  (19)  ∧E 19 :#2#2

31  DEPEND(sq,f1)⊃AXIOM(f1)  (19)  ∀E 30 f1

32  ((PROOFTREE(f cc sq)∧(PROOFTREE(sq)∧sq=scdr(f cc sq)))⊃(DEPEND(f cc sq,f1)≡DEPEND(sq,f1)))≡
  (ORI(f cc sq,sq)∨(ANDE(f cc sq,sq)∨(FALSEE(f cc sq,sq)∨(∃f.(FORM(f)∧((NOTID(f cc sq,sq,f)∨
  (NOTED(f cc sq,sq,f)∨IMPLID(f cc sq,sq,f)))∧f≠f1))∨∃x t.(INDVAR(x)∧(TERM(t)∧
  (GENI(f cc sq,sq,x,t)∨(GENE(f cc sq,sq,x,t)∨EXI(f cc sq,sq,x,t)))))))))
  ∀E DEPEND f cc sq , sq , f1

33  INDVAR(x)∧(TERM(x)∧(GENI(f cc sq,sq,x,x)∨(GENE(f cc sq,sq,x,x)∨EXI(f cc sq,sq,x,x))))
  (1 2 3 4 5 6 7 8 9 10 19)  ∧E 27 :#2

34  ∃x t.(INDVAR(x)∧(TERM(t)∧(GENI(f cc sq,sq,x,t)∨(GENE(f cc sq,sq,x,t)∨EXI(f cc sq,sq,x,t)))))
  (1 2 3 4 5 6 7 8 9 10 19)  UNIFY 33

35  DEPEND(f cc sq,f1)⊃AXIOM(f1)  (1 2 3 4 5 6 7 8 9 10 19)  1 : 34

36  ∀f1.(DEPEND(f cc sq,f1)⊃AXIOM(f1))  (1 2 3 4 5 6 7 8 9 10 19)  ∀I 35 f1 ← f1

37  f=scar(f cc sq)  (1 2 3 4 5 6 7 8 9 10 19)  1 : 36

38  PROOFTREE(f cc sq)∧(f=scar(f cc sq)∧∀f1.(DEPEND(f cc sq,f1)⊃AXIOM(f1)))
  (1 2 3 4 5 6 7 8 9 10 19)  ∧I (29 (37 36))

39  BEW(f)≡(FORM(f)∧∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)))))
  ∀E PROVABLE f

40  ∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))  (1 2 3 4 5 6 7 8 9 10)  UNIFY 38

41  BEW(f)  (1 2 3 4 5 6 7 8 9 10)  9 , 39 , 40

42  BEW(x gen f)⊃BEW(f)  (1 2 3 4 5 6 7 8 9)  ⊃I 10 41

43  BEW(f)  (43)  ASSUME

44  FORM(f)∧∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))  (43)  43 , 43 , 39

45  ∃sq.(PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))  (43)  ∧E 44 :#2
```

```
46  PROOFTREE(sq)∧(f=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1)))  (46)  ASSUME

47  ∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))  (46)  ∧E 46 :*2*2

48  DEPEND(sq,f1)⊃AXIOM(f1)  (46)  ∀E 47 f1

49  APGENI(x,sq)≡((INDVAR(x)∧∀f.(DEPEND(sq,f)⊃¬FR(x,f)))∧PROOFTREE(sq))  ∀E GENRUL2 x , sq

50  AXIOM(f1)⊃(¬FR(x,f1)∧FORM(f1))    ∀E THEORY x , f1

51  DEPEND(sq,f1)⊃¬FR(x,f1)  (1 2 3 4 5 6 7 8 9 43 46)  9 , 43 : 50

52  ∀f1.(DEPEND(sq,f1)⊃¬FR(x,f1))  (1 2 3 4 5 6 7 8 9 43 46)  ∀I 51 f1 ← f1

53  APGENI(x,sq)  (1 2 3 4 5 6 7 8 9 43 46)  9 , 43 : 52

54  GENI((x gen f) cc sq,sq,x,x) (SEQUENCE((x gen f) cc sq)∧(INDVAR(x)∧(INDVAR(x)∧
    (scdr((x gen f) cc sq)=sq∧(PROOFTREE(sq)∧∃f1.(FORM(f1)∧(scar((x gen f) cc sq)=
    (x gen f1)∧(scar(sq)=sbt(x,x,f1)∧APGENI(x,sq)))))))))
    ∀E GENRUL1  (x gen f) cc sq , sq , x , x

55  (STRING(x gen f)∧SEQUENCE(sq))⊃scar((x gen f) cc sq)=(x gen f)(2) ∀E 2 x gen f , sq

56  (STRING(x gen f)∧SEQUENCE(sq))⊃scdr((x gen f) cc sq)=sq  (3)  ∀E 3 x gen f , sq

57  (INDVAR(x)∧FORM(f))⊃FORM(x gen f)  (4)  ∀E 4 x , f

58  FORM(x gen f)⊃STRING(x gen f)  (5)  ∀E 5 x gen f

59  PROOFTREE(sq)⊃SEQUENCE(sq)  (8)  ∀E 8 sq

60  FORM(f)∧(scar((x gen f) cc sq)=(x gen f)∧(scar(sq)=sbt(x,x,f)∧APGENI(x,sq)))
    (1 2 3 4 5 6 7 8 9 43 46)  11 , 43 : 59 , 9

61  ∃f1.(FORM(f1)∧(scar((x gen f) cc sq)=(x gen f1)∧(scar(sq)=sbt(x,x,f1)∧
    APGENI(x,sq))))  (1 2 3 4 5 6 7 8 9 43 46)  UNIFY 60

62  (FORM(x gen f)∧SEQUENCE(sq))⊃SEQUENCE((x gen f) cc sq)  (6)  ∀E 6 x gen f , sq

63  GENI((x gen f) cc sq,sq,x,x)  (1 2 3 4 5 6 7 8 9 43 46)  9 , 11 , 43 : 62

64  PROOFTREE((x gen f) cc sq)≡((SEQUENCE((x gen f) cc sq)∧FORM((x gen f) cc sq))∨
    (∃pf.(PROOFTREE(pf)∧(ORI((x gen f) cc sq,pf)∨(ANDE((x gen f) cc sq,pf)∨
    (FALSEE((x gen f) cc sq,pf)∨(NOTI((x gen f) cc sq,pf)∨(NOTE((x gen f) cc sq,pf)∨
    IMPLI((x gen f) cc sq,pf)))))))∨(∃pf x1 t.(PROOFTREE(pf)∧(INDVAR(x1)∧(TERM(t)∧
    (GENI((x gen f) cc sq,pf,x1,t)∨(GENE((x gen f) cc sq,pf,x1,t)∨EXI((x gen f)
    cc sq,pf,x1,t))))))∨(∃pf1 pf2.(PROOFTREE(pf1)∧(PROOFTREE(pf2)∧(ANDI((x gen f)
    cc sq,pf1,pf2)∨(FALSEI((x gen f) cc sq,pf1,pf2)∨IMPLE((x gen f) cc sq,pf1,pf2)))))∨
    (∃pf1 pf2 x1 x2.(PROOFTREE(pf1)∧(PROOFTREE(pf2)∧(INDVAR(x1)∧(INDVAR(x2)∧EXE(
    (x gen f) cc sq,pf1,pf2,x1,x2)))))∨∃pf1 pf2 pf3.(PROOFTREE(pf1)∧(PROOFTREE(pf2)∧
    (PROOFTREE(pf3)∧ORE((x gen f) cc sq,pf1,pf2,pf3))))))))    ∀E PROOF (x gen f) cc sq

65  INDVAR(x)⊃TERM(x)  (7)  ∀E 7 x
```

66 PROOFTREE(sq)∧(INDVAR(x)∧(TERM(x)∧(GENI((x gen f) cc sq,sq,x,x)∨(GENE((x gen f) cc sq,sq,x,x)∨EXI((x gen f) cc sq,sq,x,x))))) (1 2 3 4 5 6 7 8 9 43 46) 9,43 : 65

67 ∃pf x1 t.(PROOFTREE(pf)∧(INDVAR(x1)∧(TERM(t)∧(GENI((x gen f) cc sq,pf,x1,t)∨(GENE((x gen f) cc sq,pf,x1,t)∨EXI((x gen f) cc sq,pf,x1,t))))))
(1 2 3 4 5 6 7 8 9 43 46) UNIFY 66

68 PROOFTREE((x gen f) cc sq) (1 2 3 4 5 6 7 8 9 43 46) 43 : 67 , 11 , 9

69 ((PROOFTREE((x gen f) cc sq)∧(PROOFTREE(sq)∧sq=scdr((x gen f) cc sq)))⊃(DEPEND((x gen f) cc sq,f1)≡DEPEND(sq,f1)))≡(ORI((x gen f) cc sq,sq)∨(ANDE((x gen f) cc sq,sq)∨(FALSEE((x gen f) cc sq,sq)∨(∃f.(FORM(f)∧((NOTID((x gen f) cc sq,sq,f)∨(NOTED((x gen f) cc sq,sq,f)∨IMPLID((x gen f) cc sq,sq,f)))∧f≠f1))∨∃x1 t.(INDVAR(x1)∧(TERM(t)∧(GENI((x gen f) cc sq,sq,x1,t)∨(GENE((x gen f) cc sq,sq,x1,t)∨EXI((x gen f) cc sq,sq,x1,t))))))))) ∀E DEPEND (x gen f) cc sq , sq , f1

70 INDVAR(x)∧(TERM(x)∧(GENI((x gen f) cc sq,sq,x,x)∨(GENE((x gen f) cc sq,sq,x,x)∨EXI((x gen f) cc sq,sq,x,x)))) (1 2 3 4 5 6 7 8 9 43 46) ∧E 66 :#2

71 ∃f.(INDVAR(x)∧(TERM(t)∧(GENI((x gen f) cc sq,sq,x,t)∨(GENE((x gen f) cc sq,sq,x,t)∨EXI((x gen f) cc sq,sq,x,t))))) (1 2 3 4 5 6 7 8 9 43 46) 70 x ← t OCC

72 ∃x1 t.(INDVAR(x1)∧(TERM(t)∧(GENI((x gen f) cc sq,sq,x1,t)∨(GENE((x gen f) cc sq,sq,x1,f)∨EXI((x gen f) cc sq,sq,x1,t))))) (1 2 3 4 5 6 7 8 9 43 46) 71 x←x1 OCC

73 DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1) (1 2 3 4 5 6 7 8 9 43 46) 11 , 9 , 43 : 72

74 ∀f1.(DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1)) (1 2 3 4 5 6 7 8 9 43 46) ∀I 73 f1 ← f1

75 (x gen f)=scar((x gen f) cc sq) (1 2 3 4 5 6 7 8 9 43 46) 9 , 43 : 74

76 PROOFTREE((x gen f) cc sq)∧((x gen f)=scar((x gen f) cc sq)∧∀f1.(DEPEND((x gen f) cc sq,f1)⊃AXIOM(f1))) (1 2 3 4 5 6 7 8 9 43 46) ∧I (68 (75 74))

77 BEW(x gen f)≡(FORM(x gen f)∧∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))) ∀E PROVABLE x gen f

78 ∃sq.(PROOFTREE(sq)∧((x gen f)=scar(sq)∧∀f1.(DEPEND(sq,f1)⊃AXIOM(f1))))
(1 2 3 4 5 6 7 8 9 10 19 43 46) UNIFY 76

79 BEW(x gen f) (1 2 3 4 5 6 7 8 9 43) 11 , 9 , 43 : 78

80 BEW(f)⊃BEW(x gen f) (1 2 3 4 5 6 7 8 9) ⊃I 43 79

81 BEW(x gen f)≡BEW(f) (1 2 3 4 5 6 7 8 9) ≡I 42 80

82 (INDVAR(x)∧FORM(f))⊃(BEW(x gen f)≡BEW(f)) (1 2 3 4 5 6 7 8) ⊃I 9 81

83 ∀x f.((INDVAR(x)∧FORM(f))⊃(BEW(x gen f)≡BEW(f))) (1 2 3 4 5 6 7 8) ∀I 82 x , f

84 (INDVAR(x1)∧FORM(x2 gen f))⊃(BEW(x1 gen (x2 gen f))≡BEW(x2 gen f))
(1 2 3 4 5 6 7 8) ∀E 83 x1 , x2 gen f

85 (INDVAR(x2)∧FORM(f))⊃(BEW(x2 gen f)≡BEW(f)) (1 2 3 4 5 6 7 8) ∀E 83 x2 , f

86 (INDVAR(x1)∧FORM(f))⊃(BEW(x1 gen f)≡BEW(f)) (1 2 3 4 5 6 7 8) ∀E 83 x1 , f

87 (INDVAR(x2)∧FORM(x1 gen f))⊃(BEW(x2 gen (x1 gen f))≡BEW(x1 gen f))
(1 2 3 4 5 6 7 8) ∀E 83 x2 , x1 gen f

88 (INDVAR(x1)∧FORM(f))⊃FORM(x1 gen f) (4) ∀E 4 x1 , f

89 (INDVAR(x2)∧FORM(f))⊃FORM(x2 gen f) (4) ∀E 4 x2 , f

90 (INDVAR(x1)∧(INDVAR(x2)∧FORM(f)))⊃(BEW(x1 gen (x2 gen f))≡BEW(x2 gen (x1 gen f)))
(1 2 3 4 5 6 7 8) 84 : 89

91 ∀x1 x2 f.((INDVAR(x1)∧(INDVAR(x2)∧FORM(f)))⊃(BEW(x1 gen (x2 gen f))≡BEW(x2 gen (
x1 gen f)))) (1 2 3 4 5 6 7 8) ∀I 90 x1 , x2 , f

REFERENCES

Prawitz, D.,
1965 *Natural deduction, A proof theoretical study*
Almquist and Wiksell, Stockholm (1965).

Godel, K.,
1930 *Die Vollstandingkeit der Axiome des logischen Funktionenkalkuls*
Monatshefte fur Mathematik und Physik 37, (1930) 349-360.

Godel, K.,
1931 *Uber formal unentscheidbare Satze der Principia mathematica und verwandter Systeme I*, Monatshefte fur Mathematik und Physik 38, (1931) 173-198.

Weyhrauch, R.W., and Thomas, A.J.,
1974 *FOL: A Proof Checker for First-order Logic*
Stanford Artificial Intelligence Laboratory, Memo AIM-235 (1974).